

SuperMathSX™

Coprocessor Data Sheet

SUPER MATH™

CHIPS®

9000-3786

Copyright Notice

Copyright © 1991, Chips and Technologies, Inc.

All Rights Reserved.

Printed in U.S.A.

Trademark Acknowledgement

CHIPS® and the CHIPS logotype are registered trademarks of Chips and Technologies, Inc.

Super386™, SuperMath™, SuperMathDX™, and SuperMathSX™ are trademarks of Chips and Technologies, Inc.

Intel® is a registered trademark of Intel Corporation.

Disclaimer

This manual is copyrighted by Chips and Technologies, Inc. You may not reproduce, transmit, transcribe, store in a retrieval system, or translate into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, any part of this publication without express written permission of Chips and Technologies, Inc.

The information contained in this document is being issued in advance of the production cycle for the device(s). The parameters for the device(s) may change before final production.

Chips and Technologies, Inc. makes no representations or warranties regarding the contents of this manual. We reserve the right to revise the manual or make changes in the specifications of the product described within it at any time without notice and without obligation to notify any person of such revision or changes.

The information contained in this manual is provided for general use by our customers. Our customers should be aware that the personal computer field is the subject of many patents. Our customers should ensure that they take appropriate action so that their use of our products does not infringe upon any patents. It is the policy of Chips and Technologies, Inc. to respect the valid patent rights of third parties and not to infringe upon or assist others to infringe upon such rights.

Restricted Rights and Limitations

Use, duplication, or disclosure by the Government is subject to restrictions set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at 252.277-7013

Chips and Technologies, Inc.
3050 Zanker Road
San Jose, California 95134
Phone: 408-434-0600

■ SuperMathSX™ Floating-Point Math Coprocessor

T-49-12-05

Product Brief

The SuperMathSX™ floating-point coprocessor is a state-of-the-art CMOS VLSI integrated circuit that is socket compatible and software compatible with the Intel® 80387SX™ coprocessor. SuperMathSX extends the capabilities of 80386SX compatible microprocessors by providing additional mathematical functions. It executes multiplication, division, and transcendental floating-point operations at very high speeds.

Features

The SuperMathSX coprocessor has the following features:

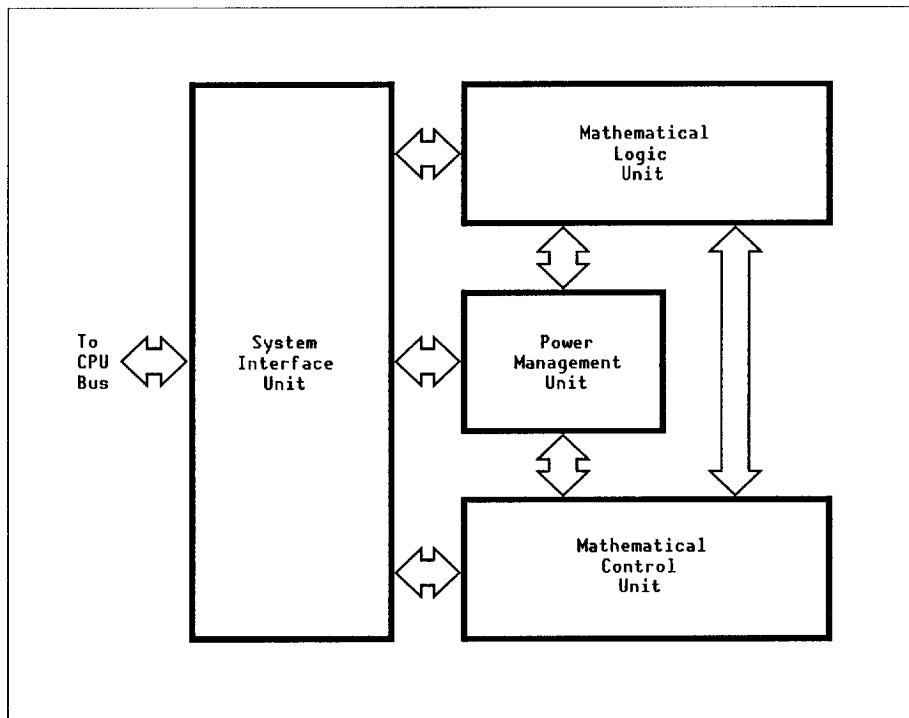
- Very high performance floating-point coprocessor for 80386SX compatible processors.
- Up to 600 percent performance improvement over the Intel 80387SX coprocessor.
- 100 percent code compatible with the 80387SX coprocessor.
- Extends the 386 SX data types to include 32-, 64-, and 80-bit floating-point operands; 32- and 64-bit integers; and 18-digit BCD operands.
- Extends the 386SX instruction set to include trigonometric, exponential, logarithmic, and arithmetic instructions.
- Complies with the ANSI/IEEE Standard 754-1985 for Binary Floating-Point Arithmetic.
- Register compatible with the 80387SX coprocessor.
- Provides eight 80-bit data registers, data register tag word, status register, and a control register.
- Available at operating speeds of up to 25MHz.
- On-chip power management circuitry reduces power consumption.
- 68-pin PLCC packaging.

CHIPS & TECHNOLOGIES INC 51E D 2098116 0002583 T1T CHP

The SuperMathSX coprocessor comprises four subsystems: the System Interface Unit, the Mathematical Logic Unit, the Mathematical Control Unit, and the Power Management Unit, as shown in the block diagram in Figure 1.

T-49-12-05

Figure 1. SuperMathSX Coprocessor Block Diagram





Contents

T-49-12-05

SuperMathSX Floating Point Math Coprocessor

Product Brief	iii
Introduction	1
System Architecture	1
Registers	2
Instruction Set	2
Accuracy	2
Interface Signals	4
Component Pin Assignment	4
Signal Descriptions	8
Bus Operation	12
ERROR* Checking Protocol	12
Instructions With No Operand Transfer	12
Instructions That Transfer Data to the SuperMathSX Coprocessor	15
Instructions That Use PEREQ to Synchronize Transfer	20
Register Set	26
Data Registers	26
Data Register Tag Word	27
Status Register	30
Control Register	33
Data Register Errors	35
Exceptions	36
Precision Exception	37
Data Underflow Exception	37
Data Overflow Exception	38
Denormal Operand Exception	39
Divide by Zero Exception	39
Invalid Operation	39

Instruction Set	41
Notation	41
F2XM1	46
FABS	48
FADD	49
FCHS	51
FCLEX	52
FCOM	53
FCOS	55
FDECSTP	57
FDIV	58
FFREE	61
FINCSTP	62
FINIT	63
FLD	64
FLD1	66
FLDCW	67
FLDENV	68
FLDL2E	70
FLDL2T	71
FLDLG2	72
FLDLN2	73
FLDPI	74
FLDZ	75
FMUL	76
FNOP	78
FPATAN	79
FPREM	81
FPREM1	83
FPTAN	85
FRNDINT	87
FRSTOR	89
FSAVE	90
FSCALE	95
FSIN	97
FSINCOS	99
FSQRT	101
FST	103
FSTCW	105
FSTENV	106
FSTSW	108
FSUB	109
FTST	111
FUCOM	113

T-49-12-05

FXAM	115
FXCH	116
FXTRACT	117
FYL2X	119
FYL2XP1	121
SuperMathSX Operands	123
AC/DC Characteristics	128
General Characteristics	128
DC Characteristics	129
AC Characteristics	130
Packaging Dimensions	134
Representatives	

List of Figures

T-49-12-05

Figure 1.	SuperMathSX Coprocessor Block Diagram	iv
Figure 2.	Memory Data Formats	3
Figure 3.	SuperMathSX Pin Assignment	6
Figure 4.	Timing for Instructions With No Operand Transfer	14
Figure 5.	Timing for Transfer of Instruction With 32-Bit Operand to the SuperMathSX Coprocessor (BUSY* Inactive)	16
Figure 6.	Timing for the Transfer of Instruction With 32-Bit Operand to the SuperMathSX Coprocessor (BUSY* Active)	17
Figure 7.	Timing for the Transfer of Instructions With 64-Bit Operand to the SuperMathSX Coprocessor (BUSY* Inactive)	18
Figure 8.	Timing for the Transfer of Instructions With 64-Bit Operand to the SuperMathSX Coprocessor (BUSY* Active)	19
Figure 9.	Timing for Transfer of Instructions That Use PEREQ to Synchronize (16-Bits)	21
Figure 10.	Timing for Transfer of Instructions That Use PEREQ to Synchronize (32-Bit)	22
Figure 11.	Timing for Transfer of Instructions That Use PEREQ to Synchronize (64-Bit)	23
Figure 12.	Timing for Transfer of Instructions That Use PEREQ to Synchronize (80-Bits or More)	24
Figure 13.	Timing for FSTCW and FSTSW Instructions	25
Figure 14.	Data Registers	26
Figure 15.	Stack Organization, Top of Stack = R0	28
Figure 16.	Stack Organization, Top of Stack = R5	29
Figure 17.	16-bit Status Register	30
Figure 18.	16-bit Control Register	33
Figure 19.	SuperMathSX Coprocessor Environment	69
Figure 20.	16-Bit Protected Mode	91
Figure 21.	16-Bit Real Mode	92
Figure 22.	32-Bit Protected Mode	93
Figure 23.	32-Bit Real Mode	94
Figure 24.	SuperMathSX Coprocessor Environment	107
Figure 25.	Data I/O Timings	131
Figure 26.	Reset Timings	132
Figure 27.	Timing After Change of ZSTEN* State	132
Figure 28.	Relative Control Signal Timings	133
Figure 29.	CPUCLK2 Waveform Characteristics	133
Figure 30.	Plastic Leaded Chip Carrier (PLCC)	134

List of Tables**T-49-12-05**

Table 1.	Assignment of Signals to Component Pins	4
Table 2.	SuperMathSX Pin Assignments	7
Table 3.	Signal Descriptions	8
Table 4.	SuperMathSX Bus Cycle Indication	11
Table 5.	Instructions With No Operand Transfer	13
Table 6.	Instructions That Transfer Data to the SuperMathSX Coprocessor	15
Table 7.	Instructions That Use PEREQ to Synchronize Transfer	20
Table 8.	Tag Field Validity	27
Table 9.	SuperMathSX Coprocessor Masked Exception Handling	37
Table 10.	Responses to Invalid Operations	40
Table 11.	Instruction Mnemonics	42
Table 12.	Argument Symbols	42
Table 13.	Opcode Locations	43
Table 14.	Field Abbreviations	43
Table 15.	Zero/Infinity Symbols	44
Table 16.	Exception Flag Symbols	45
Table 17.	FINIT Instruction Reassignments	63
Table 18.	SuperMathSX Operands	123
Table 19.	Recommended Operating Conditions	128
Table 20.	Maximum Tolerated Conditions	128
Table 21.	Capacitance	129
Table 22.	DC Characteristics	129
Table 23.	AC Operating Characteristics	130

Introduction

T-49-12-05

The SuperMathSX math coprocessor is a high-performance industry compatible floating-point mathematic coprocessor. It is pin compatible and software compatible with the industry standard 80387SX while achieving performance improvements of up to 600 percent over the 80387SX. When installed in an 80386SX-based system, the SuperMathSX coprocessor executes add, subtract, multiply, divide, and transcendental floating-point operations magnitudes faster than a standalone 80386SX.

The SuperMathSX coprocessor is code compatible with the 80387SX. Therefore, it flawlessly runs all software that has been assembled and/or compiled for the industry standard 80387SX.

In addition to maintaining industry standard compatibility, the SuperMathSX coprocessor is engineered to comply with the architecture mandated by the full extended double precision floating-point math IEEE-754-1985 specification. Full compliance with both the industry standard 80387SX and the IEEE specification guarantees that the SuperMathSX coprocessor is compatible with all current software and will be compatible with all future application software.

System Architecture

The SuperMathSX coprocessor is fabricated in a 1.2 μ double-metal CMOS process. It is available at operating speeds of up to 25MHz and is packaged in a 68-pin plastic leaded chip carrier (PLCC).

The internal structure of the SuperMathSX coprocessor can be divided into four major subsystems:

- System Interface Unit
- Mathematical Logic Unit
- Mathematical Control Unit
- Power Management Unit.

The System Interface Unit coordinates the communication and transfer of code and data between the system and the coprocessor.

The Mathematical Logic Unit performs all of the floating-point mathematic operations, including data conversions, data normalizations, and data roundings.

The Mathematical Control Unit supervises the operation of the Mathematical Logic Unit, coordinates the execution of complex calculations, and controls the data flow both to and from the System Interface Unit.

The Power Management Unit is transparent to application software. This unit monitors the activity levels within the functional block and "turns off" idle areas. This reduces power consumption and heat dissipation, positioning the SuperMathSX coprocessor well for battery-powered environments.

Registers

The SuperMathSX coprocessor is register compatible with the 80387SX. It contains eight 80-bit data registers, which are accessed in a stack fashion: a data register tag word, a status register, and a control register.

In addition, the SuperMathSX coprocessor relies on registers in the 80386SX compatible CPU, which contain pointers to:

- The memory address containing the SuperMathSX FPU's current instruction.
- The memory address containing any operand associated with the SuperMathSX FPU instruction.

Instruction Set

The binary instructions and data format used by the SuperMathSX coprocessor are fully compatible with those defined for the industry standard 80387SX. The instructions available to the user include the standard mathematic instructions, transcendental function instructions, data load and store instructions, and SuperMathSX coprocessor initialization and control instructions. Again, because these instructions are identical to the 80387SX instructions, the SuperMathSX coprocessor flawlessly runs all software that has been assembled and/or compiled for the 80387SX.

Accuracy

The SuperMathSX coprocessor supports all of the data formats mandated by the IEEE-754-1985 specification, along with other industry compatible formats. Figure 2 lists the data types and shows how they are represented in memory.

Because there is no "functional" one-to-one mapping between real numbers and the representation of real numbers in a discretely defined numbering system, the degree

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002591 096 ■ CHP

of accuracy in representing real numbers in a chosen discrete numbering system is a significant issue.

T-49-12-05

The IEEE Standard for Binary Floating-Point Arithmetic (IEEE-754-1985) specifies error bounds for floating-point calculations and values. This standard specifies both the allowable error in a given arithmetic computation and the rounding algorithms. The SuperMathSX coprocessor fully complies with the IEEE specification. Therefore, it performs calculations to the levels of accuracy specified in the IEEE document.

Figure 2. Memory Data Formats

Format	Size	Most Significant	Least Significant
Word Integer	16 bits		15 7 0 [][]
"Short" Integer	32 bits		31 22 15 7 0 [][][][]
Long Integer	63 bits	63 55 47 39 31 22 15 7 0	[][][][][][][][]
BCD Integer	80 bits	79 71 63 55 47 39 31 22 15 7 0	[S][XX][017][016][015][014][013][012][011][010][09][08][07][06][05][04][03][02][01][00]
Single Real	32 Bits		31 23 22 15 7 0 [S][Exp][Significand]
Double Real	64 Bits	63 52 51 47 39 31 22 15 7 0	[S][Exp][Significand]
Extended Real	80 bits	79 71 63 55 47 39 31 22 15 7 0	[S][Exponent][1][Significand]
Byte Displacement From Base Address		79 71 63 55 47 39 31 22 15 7 0	[+9][+8][+7][+6][+5][+4][+3][+2][+1][+0] [7...0][7...0][7...0][7...0][7...0][7...0][7...0][7...0][7...0][7...0]

CHIPS & TECHNOLOGIES INC 51E D ■ 20981116 0002592 T22 ■ CHP

Interface Signals

The SuperMathSX coprocessor maintains its high-speed data link with the 80386SX compatible processor through a signal interface that is based on standard 80386SX bus signals as well as signals dedicated specifically to the SuperMathSX coprocessor.

Component Pin Assignment

The SuperMathSX coprocessor is packaged in a 68-pin plastic leaded chip carrier (PLCC). The signal assignments to the component pins are shown in Table 1.

Table 1. *Assignment of Signals to Component Pins*

Signal Name	Pin No.	Type	Description
D15	11	I/O	Data Line 15
D14	12	I/O	Data Line 14
D13	15	I/O	Data Line 13
D12	16	I/O	Data Line 12
D11	30	I/O	Data Line 11
D10	29	I/O	Data Line 10
D9	28	I/O	Data Line 9
D8	24	I/O	Data Line 8
D7	2	I/O	Data Line 7
D6	3	I/O	Data Line 6
D5	6	I/O	Data Line 5
D4	7	I/O	Data Line 4
D3	8	I/O	Data Line 3
D2	23	I/O	Data Line 2
D1	20	I/O	Data Line 1
D0	19	I/O	Data Line 0
ADS*	47	I	Address Strobe
BUSY*	36	O	Coprocessor Busy
CKM	59	N/C	Clock Mode Select (ignored by the SuperMathSX coprocessor)
CPUCLK2	54	I	CPU Clock
CMD0*	48	I	Command
ERROR*	35	O	Error

CHIPS & TECHNOLOGIES INC 51E D ■ 2098114 0002593 969 ■ CHP

Table 1. Assignment of Signals to Component Pins (continued)

T-49-12-05

Signal Name	Pin No.	Type	Description
NPCS1*	44	I	Chip Select 1
NPCS2	45	I	Chip Select 2
NPCLK2	53	N/C	Alternate Clock (ignored by the SuperMathSX coprocessor)
PEREQ	56	O	Processor Extension Request
READY*	49	I	System Ready
READYO*	57	O	Ready Output
RESETIN	51	I	Reset
W/R*	41	I	Write/Read
ZSTEN*	40	I	Z-State Enable
Vcc			Pins 4, 9, 13, 22, 26, 31, 33, 37, 43, 46, and 58
Vss			Pins 5, 14, 21, 25, 27, 32, 34, 38, 42, 55, 60, 61, 63, and 66
No connect			Pins 1, 10, 17, 18, 52, 65, 67, 68
Vcc pull-ups			Pins 39 and 50

Figure 3 shows the Pin assignment for the SuperMathSX coprocessor.

T-49-12-05

NC	1	68	INC	51	RESETIN
D7	2	67	INC	50	Pull-up to Vcc
D6	3	66	Vss	49	READY*
Vcc	4	65	INC	48	CMD0*
Vss	5	64	Vcc	47	ADS*
D5	6	63	Vss	46	Vcc
D4	7	62	Vcc	45	NPCS2
D3	8	61	Vss	44	NPCS1*
Vcc	9	60	Vss	43	Vcc
NC	10	59	CKM	42	Vss
D15	11	58	Vcc	41	W/R*
D14	12	57	READY0*	40	ZSTEN*
Vcc	13	56	PEREQ	39	Pull-up to Vcc
Vss	14	55	Vss	38	Vss
D13	15	54	CPCLK2	37	Vcc
D12	16	53	INPUCLK2	36	BUSY*
NC	17	52	INC	35	ERROR*
NC	18				
D0	19				
D1	20				
Vss	21				
Vcc	22				
D2	23				
D8	24				
Vss	25				
Vcc	26				
Vss	27				
D9	28				
D10	29				
D11	30				
Vcc	31				
Vss	32				
Vcc	33				
Vss	34				

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002595 731 ■ CHP

Table 2 lists the SuperMathSX Pin Assignments.

T-49-12-05

Table 2. *SuperMathSX Pin Assignments*

Signal Name	Pin No.	Signal Name	Pin No.	Signal Name	Pin No.
NC	1	D8	24	ADS*	47
D7	2	Vss	25	CMD0*	48
D6	3	Vcc	26	READY*	49
Vcc	4	Vss	27	Pull-up to Vcc	50
Vss	5	D9	28	RESETIN	51
D5	6	D10	29	NC	52
D4	7	D11	30	NPUCCLK2	53
D3	8	Vcc	31	CPUCCLK2	54
Vcc	9	Vss	32	Vss	55
NC	10	Vcc	33	PEREQ	56
D15	11	Vss	34	READYO*	57
D14	12	ERROR*	35	Vcc	58
Vcc	13	BUSY*	36	CKM	59
Vss	14	Vcc	37	Vss	60
D13	15	Vss	38	Vss	61
D12	16	Pull-up to Vcc	39	Vcc	62
NC	17	ZSTEN*	40	Vss	63
NC	18	W/R*	41	Vcc	64
D0	19	Vss	42	NC	65
D1	20	Vcc	43	Vss	66
Vss	21	NPCS1*	44	NC	67
Vcc	22	NPCS2	45	NC	68
D2	23	Vcc	46		

Signal Descriptions

T-49-12-05

Signal descriptions are summarized in Table 3.

Table 3. *Signal Descriptions*

Signal Name	Pin No.	Type	Description
Control and Data Bus Signals			
ADS*	47	I	Address Strobe An input that qualifies the control input signals (CMD0*, NPCS1*, NPCS2, and W/R*) from the system. After ADS* goes active, the SuperMathSX coprocessor samples the control signals. This signal is normally connected to the CPU's ADS* output.
CMD0*	48	I	Command An input that distinguishes between a command port access (CMD0* active) and a data port access (CMD0* inactive). ADS* and CPUCLK2 qualify CMD0* as valid. This signal is normally connected to the CPU's A2 address line.
NPCS1*	44	I	Chip Select 1 This input, along with NPCS2, indicates that the system is selecting the SuperMathSX coprocessor for data transfer. ADS*, CPUCLK2, and NPCS2 qualify NPCS1* as valid. This signal is normally connected to the CPU's M/IO* output.
NPCS2	45	I	Chip Select 2 This input, along with NPCS1*, indicates that the system is selecting the SuperMathSX coprocessor for data transfer. ADS*, CPUCLK2, and NPCS1* qualify NPCS2 as valid. This signal is normally connected to the CPU's A31 address line.
W/R*	41	I	Write/Read An input that distinguishes between a read and write bus cycle. ADS* and CPUCLK2 qualify W/R* as valid. This signal is normally connected to the CPU's W/R* output.

Table 3. Signal Descriptions (continued)

T-49-12-05

Signal Name	Pin No.	Type	Description
Control and Data Bus Signals, continued			
D15	11	I/O	16-Bit Data Bus This bus transfers commands, data, and status information between the system and the SuperMathSX coprocessor. CPUCLK2 qualifies the data on these lines as valid; the control input signals (CMD0*, NPCS1*, and W/R*) indicate the type of information. This bus is normally connected to the CPU's data bus (D15:0).
D14	12	I/O	
D13	15	I/O	
D12	16	I/O	
D11	30	I/O	
D10	29	I/O	
D9	28	I/O	
D8	24	I/O	
D7	2	I/O	
D6	3	I/O	
D5	6	I/O	
D4	7	I/O	
D3	8	I/O	
D2	23	I/O	
D1	20	I/O	
D0	19	I/O	
Clock Signals			
CKM	59	N/C	Clock Mode Select If an input is connected to this signal, it will be ignored by the SuperMathSX coprocessor. This signal is normally connected to Vcc.
CPUCLK2	54	I	CPU Clock This input provides the clock synchronization between the 80386SX-compatible CPU and the SuperMathSX coprocessor. This clock also provides the timing for internal operations. CPUCLK2 must be connected to the same clock source used by the CPU.
NPUCLK2	53	N/C	Alternate Clock If an input is connected to this signal, it will be ignored by the SuperMathSX coprocessor. This signal is normally left unconnected for typical applications.

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002598 440 ■ CHP

Table 3. *Signal Descriptions (continued)*

T-49-12-05

Signal Name	Pin No.	Type	Description
SuperMathSX Status Signals			
BUSY*	36	O	Coprocessor Busy This signal's output indicates that the SuperMathSX coprocessor is busy executing an instruction. The status of BUSY* is valid only when qualified by CPUCLK2. CPUCLK2 is typically connected to either the CPU's BUSY* input or the chipset's BUSY* input.
ERROR*	35	O	Error This output indicates the status of the ES bit in the Status Register. ERROR* is valid before the SuperMathSX coprocessor sets BUSY* inactive. ERROR* is also set active immediately after a system reset (to indicate the presence of the SuperMathSX coprocessor to the CPU). This signal is typically connected to either the CPU's ERROR* input or the chipset's ERROR* input.
PEREQ	56	O	Processor Extension Request This output indicates that the SuperMathSX coprocessor requires the CPU to perform a data transfer. PEREQ is active for as long as the SuperMathSX coprocessor requires further data transfers. PEREQ can only be active when BUSY* is active. This signal is normally connected to the CPU's PEREQ input.
RESETIN	51	I	SuperMathSX Reset This input instructs the SuperMathSX coprocessor to reset itself. RESETIN must remain active for at least 20 CPUCLK2 cycle periods and must be synchronized with CPUCLK2 when it goes inactive. After RESETIN goes inactive, the first SuperMathSX coprocessor access must be delayed by at least 8 CPUCLK2 cycle periods. During reset, ERROR* is set to active (to indicate the presence of the SuperMathSX coprocessor to the CPU). This signal is normally connected to the same source that drives the CPU's reset input.

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002599 387 ■ CHP

Table 3. *Signal Descriptions (continued)*

T-49-12-05

Signal Name	Pin No.	Type	Description
Miscellaneous SuperMathSX Signals			
READY*	49	I	System Ready When active, this signal, along with ADS*, indicates to the SuperMathSX coprocessor that the CPU's bus cycle is about to terminate. This signal is normally connected to the same source that drives the CPU's READY* input.
READYO*	57	O	Ready Output This output indicates that the SuperMathSX coprocessor is about to complete a bus cycle. This signal is typically connected to either the CPU's READY* input or the chipset's READY* input.
ZSTEN*	40	I	Z-State Enable When active, this pin tri-states the SuperMathSX coprocessor's BUSY*, ERROR*, PEREQ, READYO*, and D15:0 outputs. (Also, during a ZSTEN* active input, all other inputs are ignored.) Pulling this input low isolates the SuperMathSX coprocessor from the rest of the board circuitry, facilitating board-level testing. This signal is normally pulled-up to Vcc through a resistor.

Table 4 shows the values of CMD0* and W/R* during various bus cycles.

Table 4. *SuperMathSX Coprocessor Bus Cycle Indication*

Bus Cycle ¹	CMD0*	W/R*
Control/Status Read from the SuperMathSX coprocessor	0	0
Write Opcode to the SuperMathSX coprocessor	0	1
Read Data from the SuperMathSX coprocessor	1	0
Write Data to the SuperMathSX coprocessor	1	1

¹ If NPS1* = 0 and NPS2 = 1 is not true, the bus cycle is not for the SuperMathSX coprocessor and the CMD0* and W/R* signal states are don't care. Also, ZSTEN* must be inactive (=1).

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002600 929 ■ CHP

Bus Operation

T-49-12-05

The SuperMathSX coprocessor supports the numeric coprocessor interface provided by 80386SX-compatible CPUs running in both pipelined and nonpipelined formats. The SuperMathSX coprocessor performs data exchanges with the CPU with wait state accesses.

The system interface activity depends on the instruction being executed. The protocol varies considerably for different types of instructions. Generally, the SuperMathSX coprocessor performs handshaking with the CPU through the BUSY*, ERROR* and PEREQ signals. The CPU provides dedicated pins to accommodate these signals. In this way, the SuperMathSX coprocessor can generate the READYO* signal immediately after receiving the ADS* signal, thus freeing the CPU buses for DMA cycles while the SuperMathSX coprocessor is executing an instruction.

ERROR* Checking Protocol

Before initiating any SuperMathSX FPU operation, the CPU must check the ERROR* signal according to a specific protocol. The CPU first waits for BUSY* to go inactive, then checks ERROR* and proceeds as follows:

- If the CPU sees ERROR* active, it generates a coprocessor exception and executes the appropriate routine.
- If the CPU sees ERROR* inactive, it writes the instruction opcode or data to the SuperMathSX coprocessor and continues program execution.

The descriptions which follow refer to the ERROR* checking protocol.

Instructions With No Operand Transfer

For the most basic SuperMathSX FPU instructions, no external information transfer is required. Table 5 lists these instructions.

The CPU follows the ERROR* checking protocol described above to write the instruction opcode. Once the opcode is written, the SuperMathSX coprocessor sets BUSY* active and begins executing the instruction. If an exception occurs and the exception is enabled, the SuperMathSX coprocessor sets ERROR* active. This type of operation occurs in only one bus cycle and does not involve the PEREQ signal.

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002601 865 ■ CHP

Two instructions, FINIT and FCLEX, do not require the CPU to check BUSY*. The CPU can send its opcode regardless of the state of BUSY* (although the SuperMathSX coprocessor will set BUSY* active upon receipt of this instruction).

T-49-12-05

Table 5. *Instructions With No Operand Transfer*

Instruction	Instruction	Instruction
F2XM1	FLDL2E	FSINCOS
FABS	FLDL2T	FSQRT
FADD ¹	FLDLG2	FSUB ¹
FCHS	FLDLN2	FSUBR ¹
FCOM ¹	FLDPI	FTST
FCOMP	FLDZ	FUCOM
FCOMPP	FMUL ¹	FUCOMP
FCOS	FNOP	FUCOMPP
FDECSTP	FPATAN	FXAM
FDIV ¹	FPREM	FXCH
FDIVR ¹	FPREM1	EXTRACT
FFREE	FPTAN	FYL2X
FINCSTP	FRNDINT	FYL2XP1
FLD ¹	FSCALE	
FLD1	FSIN	

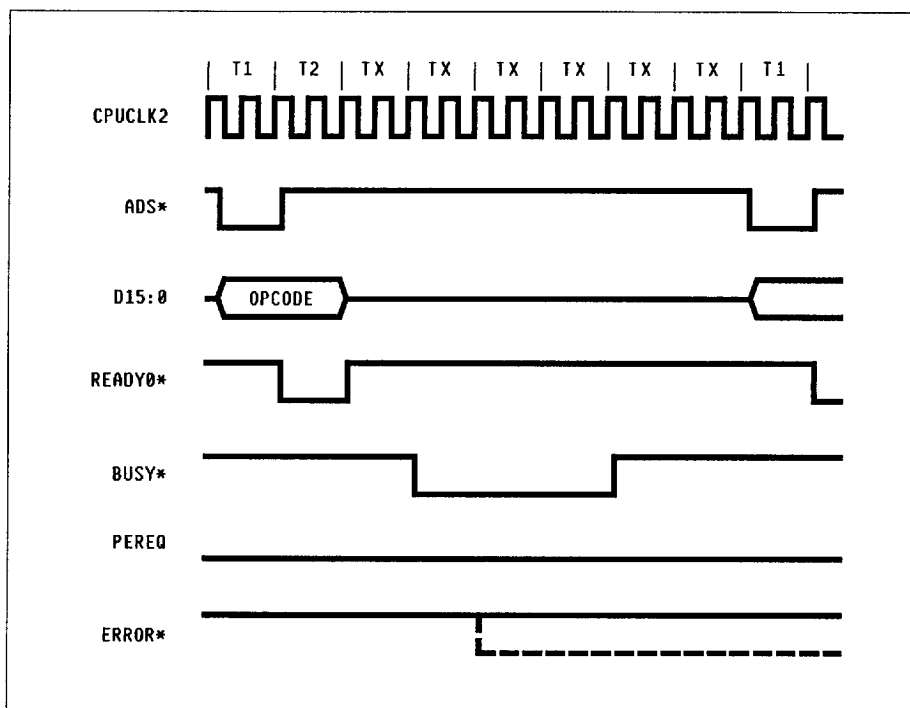
¹ Register-to-register operations only.

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002602 7T1 ■ CHP

Figure 4 shows an example of the timing for this type of operation. FINIT executes in the manner depicted in Figure 4.

T-49-12-05

Figure 4. Timing for Instructions With No Operand Transfer



Example Instructions:

FADD	reg,reg	FFREE	FCOS	FSCALE	FABS
FCOM	reg,reg	FLD1	FDECSTP	FSIN	FCHS
FDIV	reg,reg	FLDL2E	FINCSTP	FSINCOS	FNOP
FMUL	reg,reg	FLDLTE	FPATAN	FSQRT	FTST
FSUB	reg,reg	FLDLG2	FPREM	FXTRACT	FXAM
FUCOMP	reg,reg	FLDLN2	FPREM1	FYL2X	
FLD	reg,reg	FLDPI	FPTAN	FYL2XP1	
FST	reg,reg	FLDZ	FRNDINT	F2XM1	
FXCH	reg,reg				

Instructions That Transfer Data to the SuperMathSX Coprocessor

T-49-12-05

Certain instructions require a single operand transfer from the CPU to the SuperMathSX coprocessor. Table 6 lists these instructions. The CPU need not wait for BUSY* to go inactive in this case; it simply writes the instruction opcode to the SuperMathSX microprocessor. It then must wait for BUSY* to go inactive and follow the ERROR* checking protocol described above before writing the operand. This operation requires two bus cycles for 32-bit operands, and four bus cycles for 64-bit operands.

Once the operand is written, the SuperMathSX coprocessor sets BUSY* active and begins executing the instruction. If an exception occurs and is enabled, the SuperMathSX coprocessor sets ERROR* active. The PEREQ line is not used.

Table 6. *Instructions That Transfer Data to the SuperMathSX Coprocessor*

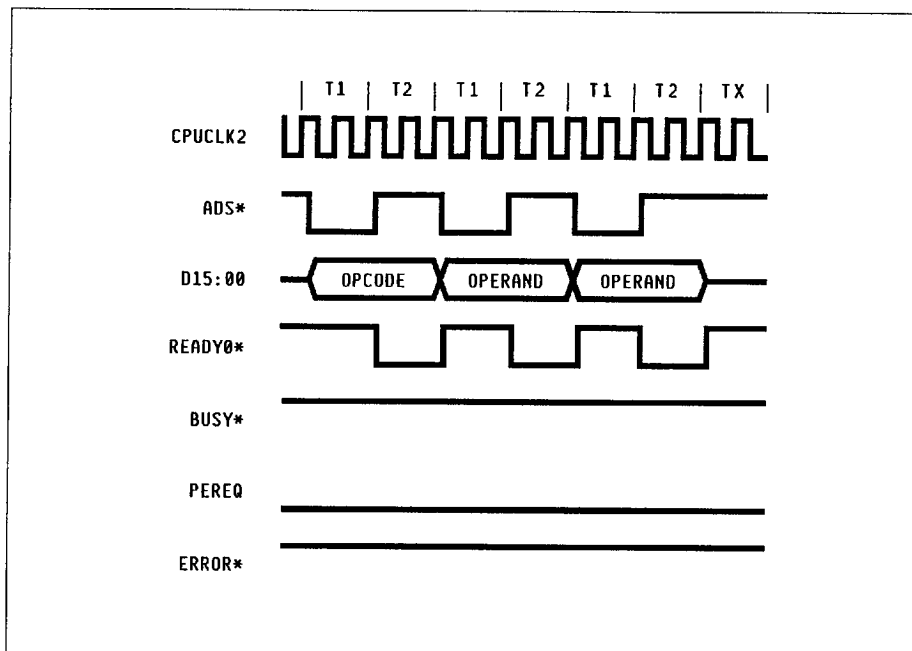
Instruction	Instruction	Instruction
FLD ¹	FCOMP	FMUL
FADD	FDIV	FSUB
FCOM	FDIVR	FSUBR

¹ 32-bit and 64-bit transfers only.

Figures 5, 6, 7, and 8 show an example of the timing for this type of operation.

T-49-12-05

Figure 5. *Timing for Transfer of Instruction With 32-Bit Operand to the SuperMathSX Coprocessor (BUSY* Inactive)*

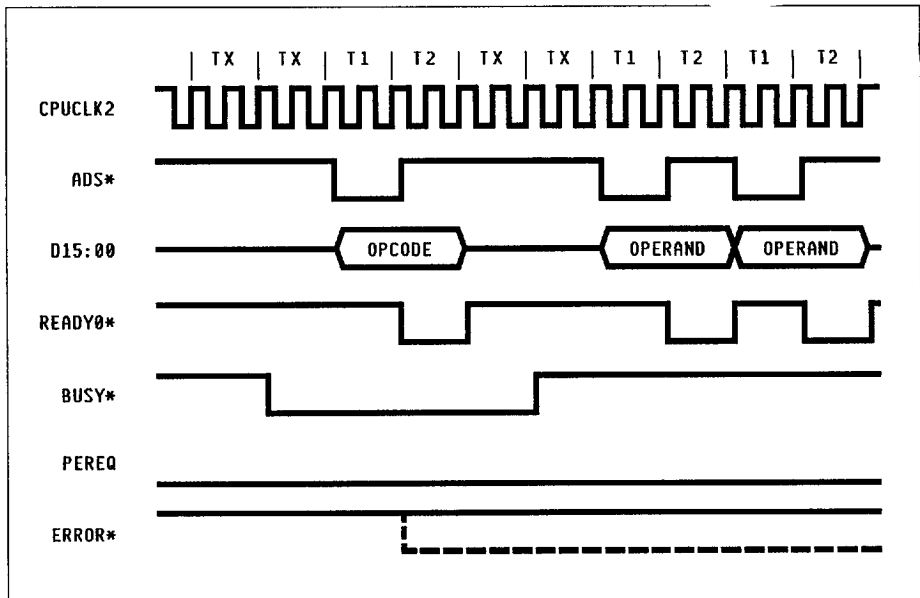


Example Instructions:

FLD	reg,mem (32-bit real)	FDIV	reg,mem (32-bit real)
FILD	reg,mem (32-bit integer)		reg,mem (32-bit integer)
FADD	reg,mem (32-bit real)	FDIV	reg,mem (32-bit real)
	reg,mem (32-bit integer)		reg,mem (32-bit integer)
		FMUL	reg,mem (32-bit real)
			reg,mem (32-bit integer)
		FSUB	reg,mem (32-bit real)
			reg,mem (32-bit integer)

Figure 6. Timing for Transfer of Instruction With 32-Bit Operand to the SuperMathSX Coprocessor (*BUSY** Active)

T-49-12-05



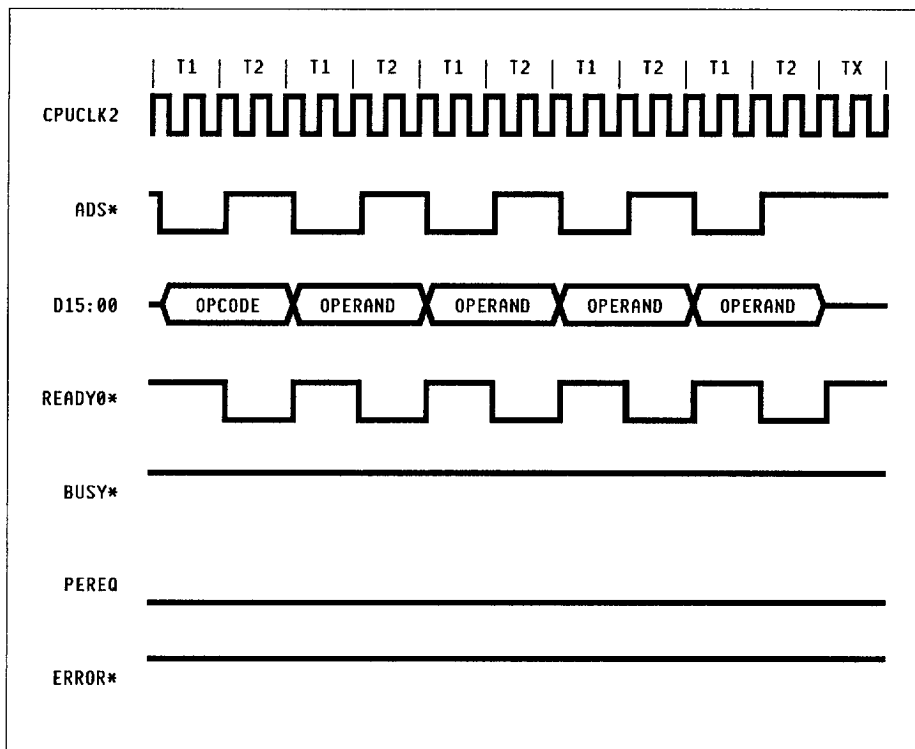
Example Instructions:

FLD	reg,mem (32-bit real)	FDIV	reg,mem (32-bit real)
FILD	reg,mem (32-bit integer)	FDIV	reg,mem (32-bit integer)
FADD	reg,mem (32-bit real)	FDIV	reg,mem (32-bit real)
	reg,mem (32-bit integer)	FMUL	reg,mem (32-bit integer)
		FMUL	reg,mem (32-bit real)
		FSUB	reg,mem (32-bit integer)
		FSUB	reg,mem (32-bit real)

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002606 347 ■ CHP

Figure 7. *Timing for the Transfer of Instructions With 64-Bit Operand to the SuperMathSX Coprocessor (BUSY* Inactive)*

T-49-12-05



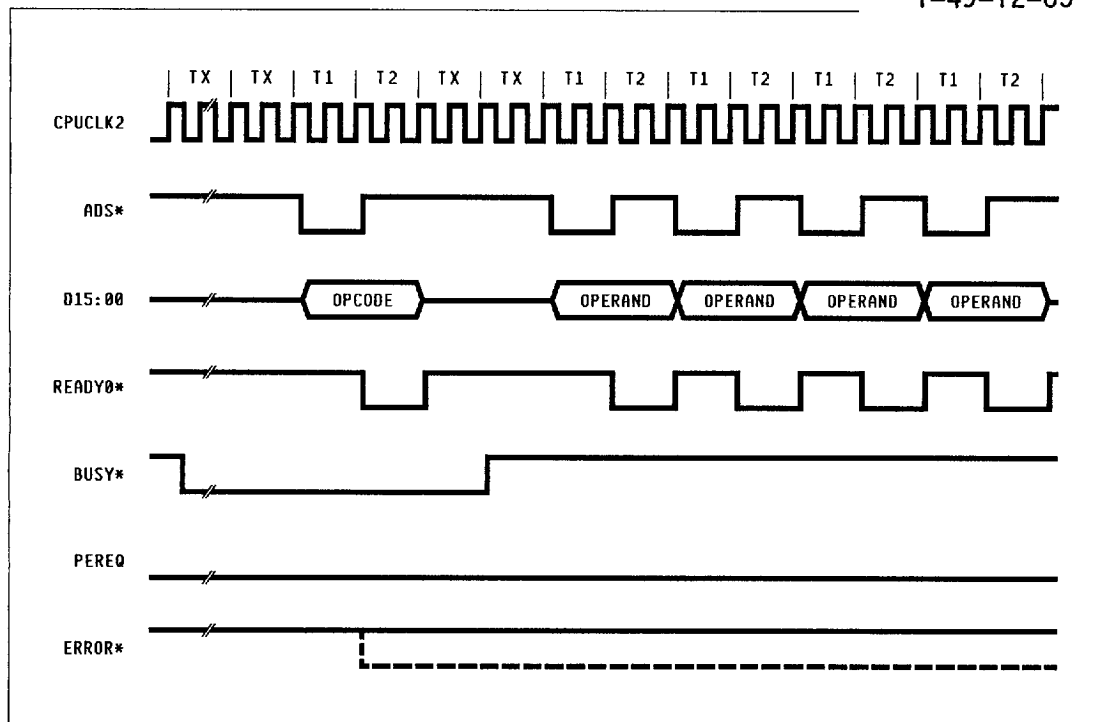
Example Instructions:

FLD	reg.mem (64-bit real)	FCOM	reg.mem (64-bit real)
FILD	reg.mem (64-bit integer)		reg.mem (64-bit integer)
FADD	reg.mem (64-bit real)	FDIV	reg.mem (64-bit real)
	reg.mem (64-bit integer)		reg.mem (64-bit integer)
		FMUL	reg.mem (64-bit real)
			reg.mem (64-bit integer)
		FSUB	reg.mem (64-bit real)
			reg.mem (64-bit integer)

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002607 283 ■ CHP

Figure 8. Timing for the Transfer of Instructions With 64-Bit Operand to the SuperMathSX Coprocessor (*BUSY** Active)

T-49-12-05



Example Instructions:

FLD	reg,mem (64-bit real)	FCOM	reg,mem (64-bit real)
FILD	reg,mem (64-bit integer)		reg,mem (64-bit integer)
FADD	reg,mem (64-bit real)	FDIV	reg,mem (64-bit real)
	reg,mem (64-bit integer)		reg,mem (64-bit integer)
		FMUL	reg,mem (64-bit real)
			reg,mem (64-bit integer)
		FSUB	reg,mem (64-bit real)
			reg,mem (64-bit integer)

Instructions That Use PEREQ to Synchronize Transfer

The instructions that return an operand from the SuperMathSX coprocessor, as well as instructions that write 16-bit and 80-bit operands to the SuperMathSX coprocessor, must synchronize the operand transfer with the PEREQ signal. Table 7 lists these instructions.

The CPU follows the ERROR* checking protocol described above to write the instruction opcode. It then waits for both BUSY* and PEREQ to go active before transferring the operand. BUSY* stays active, and PEREQ inactive, while the operand is converted to the proper format. PEREQ then goes active when the value is ready for transfer.

This operation requires 1 bus cycle to transfer a 16-bit word integer; 2 bus cycles for 32-bit short integer and 32-bit single-precision real operands; 4 bus cycles to transfer a 64-bit long integer and 64-bit double precision real operands; 5 bus cycles to transfer 80-bit operands; and various values for special operations as noted in Table 7.

Table 7. *Instructions That Use PEREQ to Synchronize Transfer*

Instruction	Bus Cycles	Instruction	Bus Cycles
FLD ¹	6	FIDIVR ²	2
FILD ²	2	FISUBR ²	2
FICOMP ²	2	FLDCW	2
FIMUL ²	2	FLDENV	8
FIADD ²	2	FRSTOR	48
FIDIV ²	2	FSAVE	48
FISUB ²	2	FST	2/3/4
FICOM ²	2	FSTENV	8

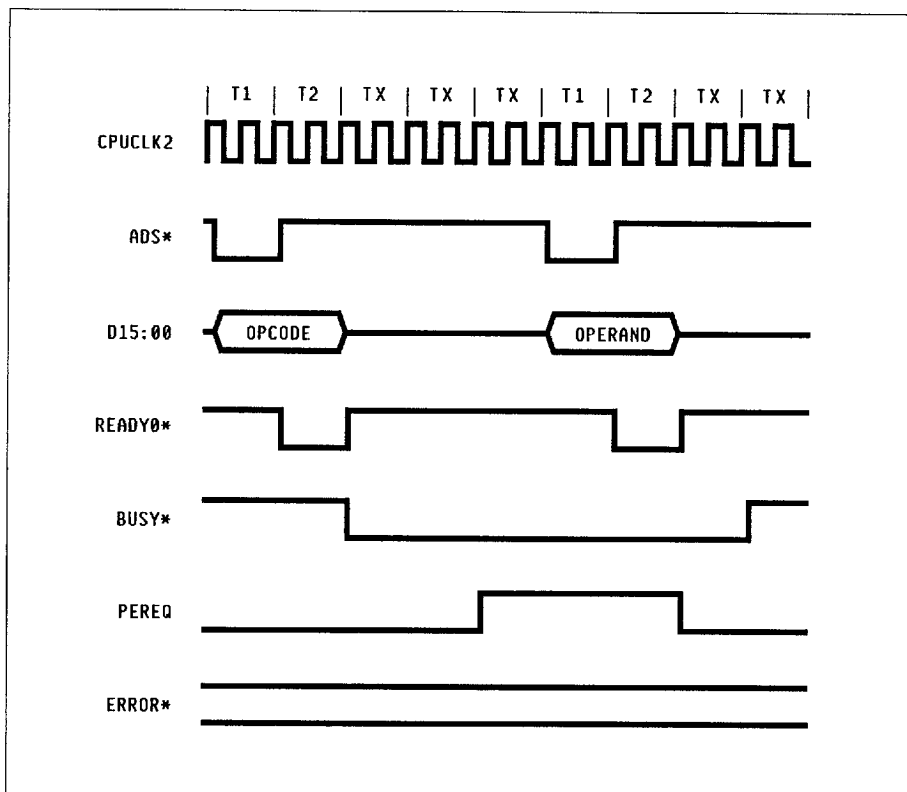
¹ 80-bit operands only.

² 16-bit integers only.

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002609 056 ■ CHP

Figures 9, 10, 11, and 12 show examples of the timing for this type of operation.

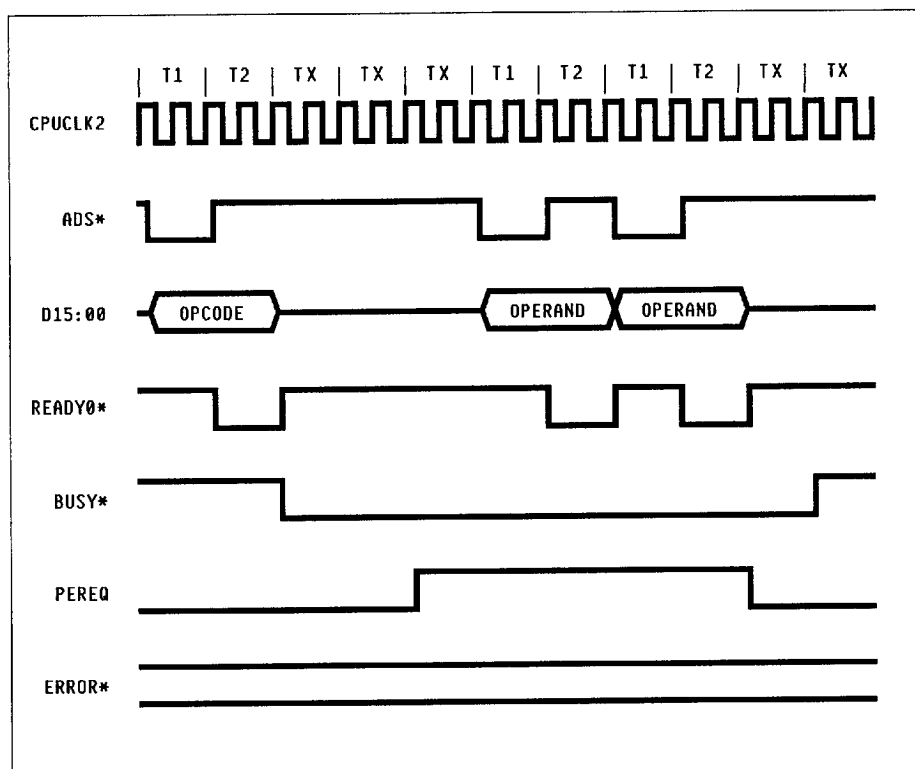
T-49-12-05

Figure 9. Timing for Transfer of Instructions That Use PEREQ to Synchronize (16-Bit)**Example Instructions:**

FILD	reg,mem (16-bit integer)	FICOM	reg,mem (16-bit integer)
FLDCW		FIDIV	reg,mem (16-bit integer)
FIST	reg,mem (16-bit integer)	FIMUL	reg,mem (16-bit integer)
FIADD	reg,mem (16-bit integer)	FISUB	reg,mem (16-bit integer)

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002610 878 ■ CHP

Figure 10. *Timing for Transfer of Instructions That Use PEREQ to Synchronize (32-Bit)*



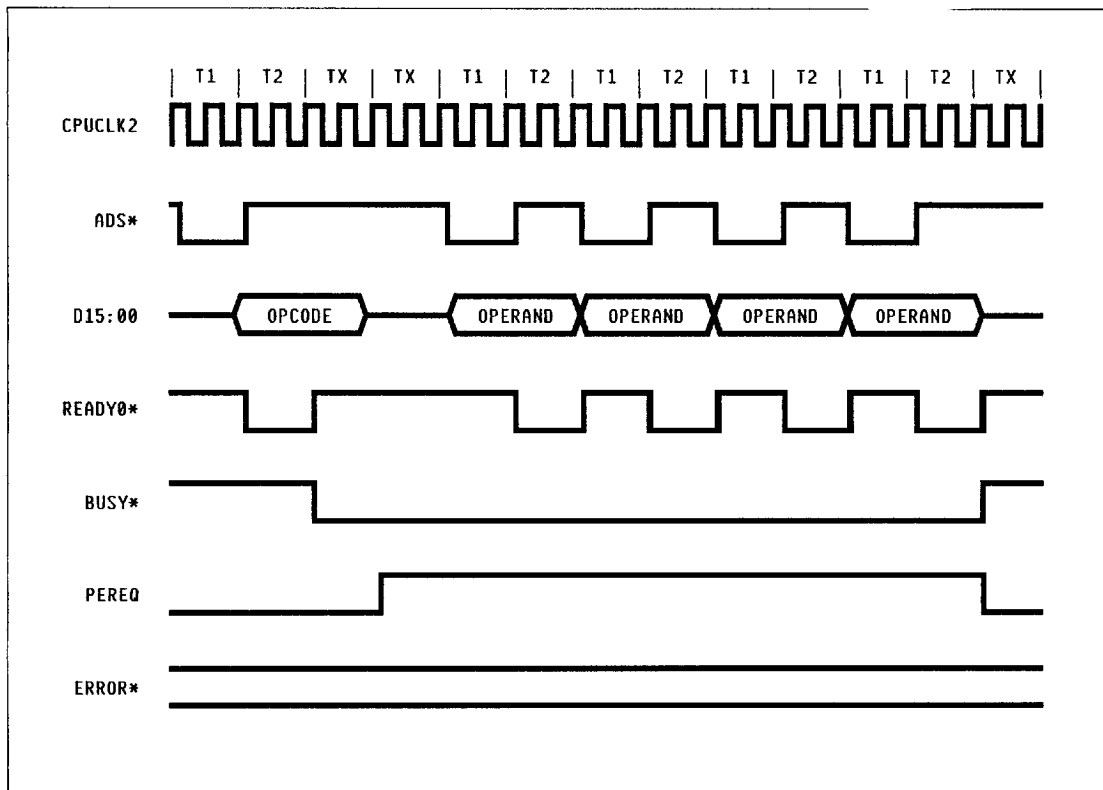
Example Instructions:

FST reg,mem (32-bit real)
FIST reg,mem (32-bit integer)

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002611 704 ■ CHP

Figure 11. *Timing for Transfer of Instructions That Use PEREQ to Synchronize (64-Bit)*

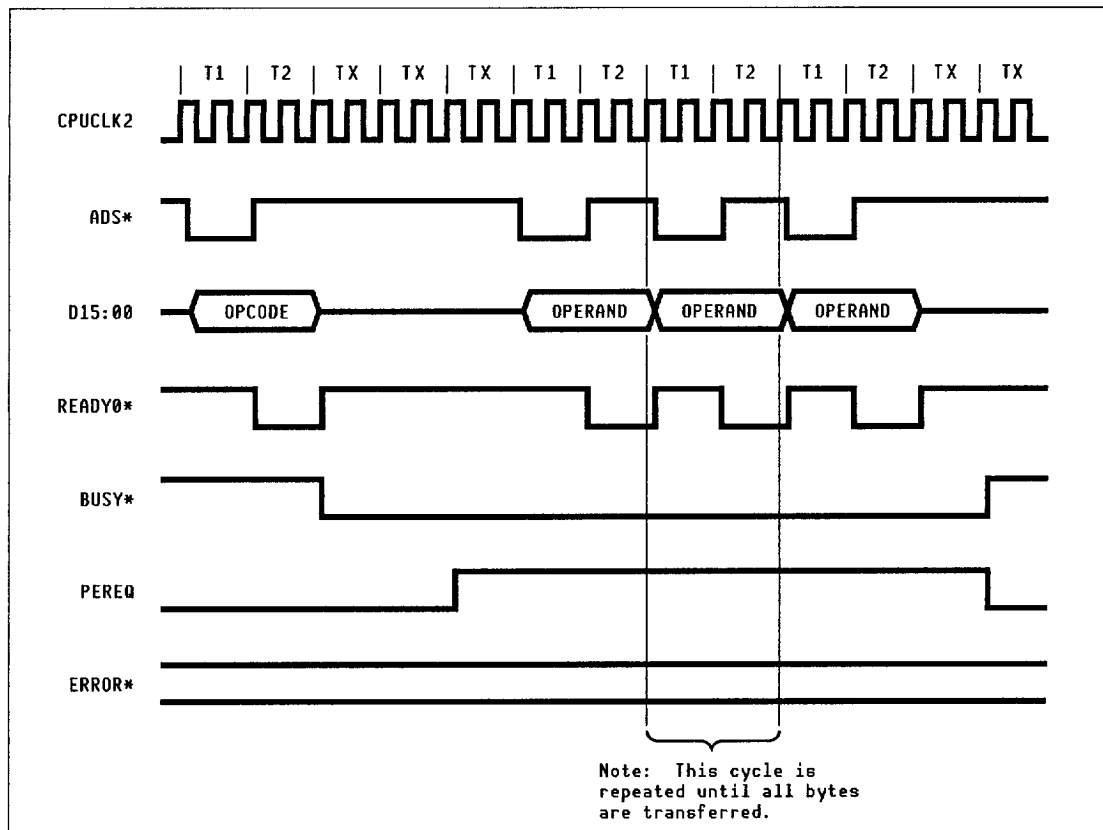
T-49-12-05

**Example Instructions:**

FST reg,mem (64-bit real)
 FIST reg,mem (64-bit integer)

Figure 12. *Timing for Transfer of Instructions That Use PEREQ to Synchronize (80 Bits or More)*

T-49-12-05



Example Instructions:

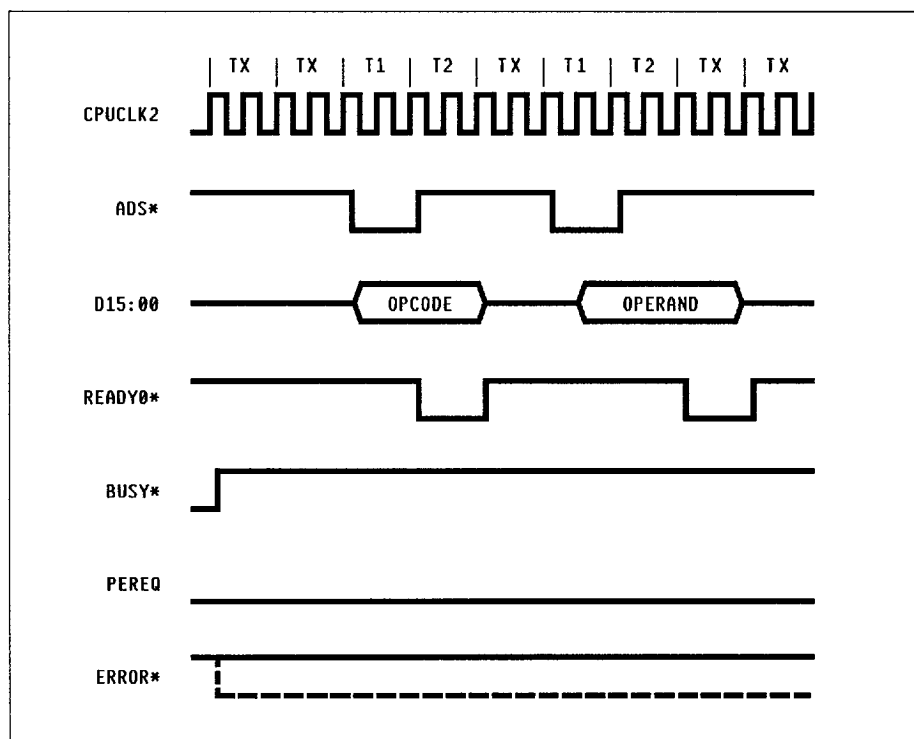
FLD reg,mem (80-bit real)
 FBLD reg,mem (80-bit BCD)
 FLDENV
 FSTENV

FST reg,mem (80-bit real)
 FBSTP reg,mem (80-bit BCD)
 FRSTOR
 FSAVE

Two instructions, FSTCW and FSTSW, perform a return of instantly available data to the CPU from the SuperMathSX coprocessor. Since there is no calculation required on the part of the SuperMathSX coprocessor, the CPU only has to follow the ERROR* checking protocol in waiting for BUSY* to go inactive (see "Error Checking Protocol"). The CPU can then write the instruction opcode and immediately read the requested data without waiting for PEREQ. These instructions take two bus cycles to complete.

Figure 13 shows an example of the timing for this type of operation.

Figure 13. Timing for FSTCW and FSTSW Instructions



Example Instructions:

FSTCW

FSTSW
FSTSWAX

Register Set

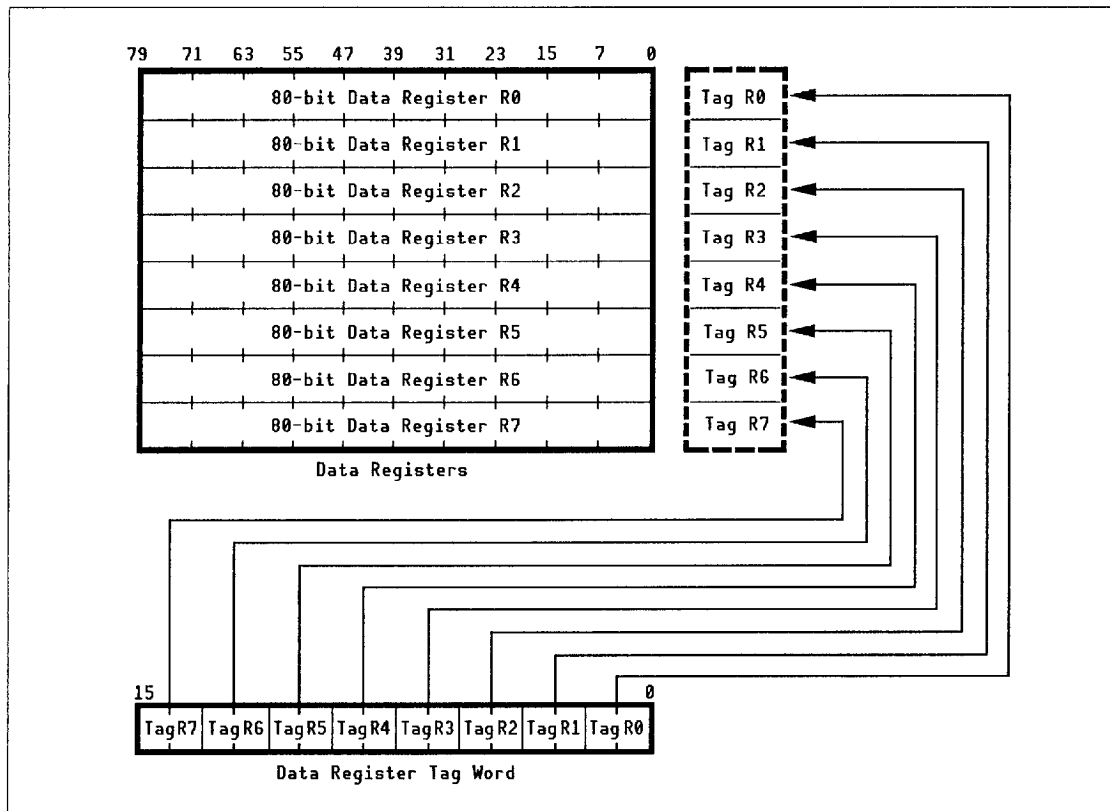
T-49-12-05

The SuperMathSX coprocessor provides eight 80-bit data registers, which are accessed in a stack-like fashion. In addition, there is a Data Register Tag Word, a Status Register, and a SuperMathSX Control Register. The SuperMathSX coprocessor is register compatible with the Intel 80387SX coprocessor.

Data Registers

The eight 80-bit data registers serve for all SuperMath computations (Figure 14). These registers are either accessed in stack-like fashion or are individually addressed.

Figure 14. Data Registers



CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002615 35T ■ CHP

As a stack, the SuperMathSX FPU instruction set can perform operations on the value in the register designated as the Top of Stack and either a value located in system memory or a value residing in one of the SuperMathSX coprocessor's other seven data registers. The register designated as the Top of Stack is selected by the TS2:0 bit field in the Status Register. The remaining data registers are then addressed in unit incremental steps (with a rollover occurring from Data Register R7 to Data Register R0) relative to the chosen Top of Stack register (see Figures 14 and 15).

After a RESET sequence, the values in the data registers are undefined. Execution of a FINIT (initialize) instruction does not change the values in the data registers. However, each data register tag field defaults to a binary value of 11 (Data Register Empty).

Data Register Tag Word

Each data register has a corresponding tag field (Figures 15 and 16). The eight tag fields are contained in the Data Register Tag Word. The tag field for each data register indicates the validity of the contents in the data register, in accordance with Table 8.

Table 8. *Tag Field Validity*

B1	B0	Data Register Contents
0	0	Register contents valid.
0	1	Register contents value equals zero.
1	0	Register contents are either QNaN, SNaN, Infinity, Denormal, or in an unsupported format.
1	1	Register empty (default after RESET/FINIT).

Note that the tag fields are not Top of Stack relative. Each tag field corresponds to a specific data register, regardless of the Top of Stack designated by the TS2:0 bit field in the Status Register. (See Figure 15.) Software should read the value in the TS2:0 bit field to determine which tag field in the Data Register Tag Word corresponds to the Top of Stack.

The Data Register Tag Word defaults to a value of FFh at the completion of either a RESET sequence or FINIT instruction.

Figure 15. Stack Organization, Top of Stack = R0

T-49-12-05

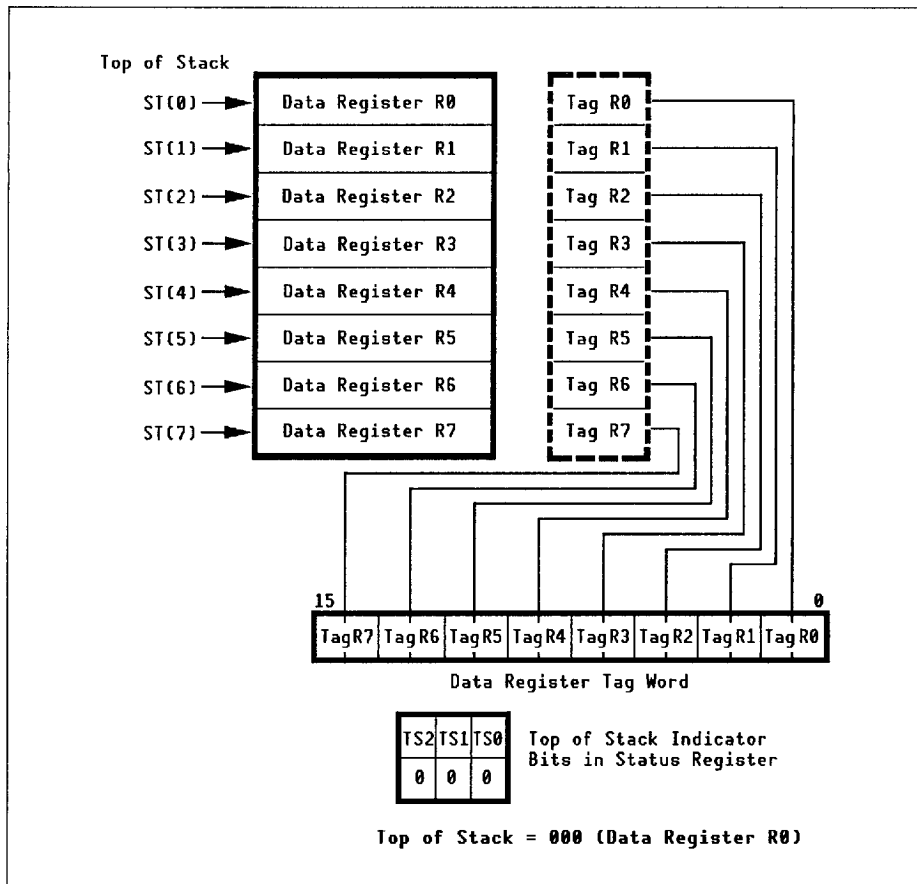
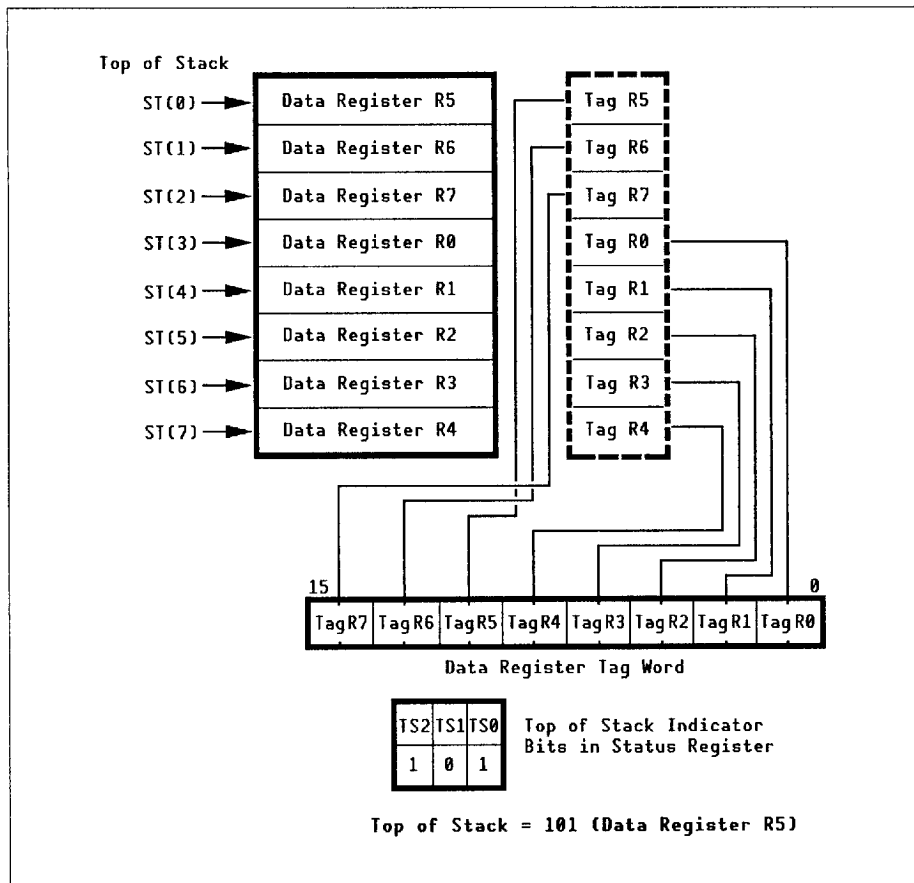


Figure 16. Stack Organization, Top of Stack = R5

T-49-12-05

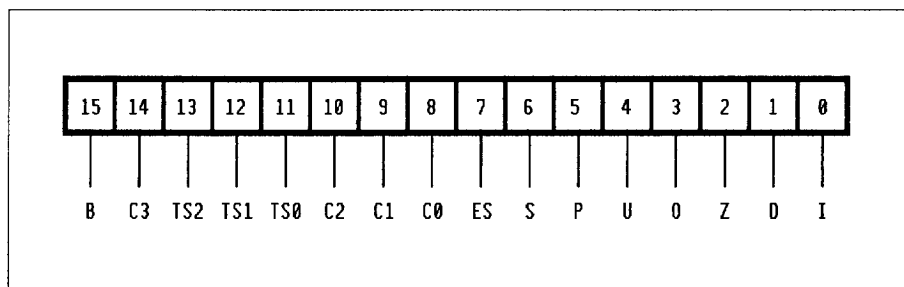


Status Register

T-49-12-05

The 16-bit Status Register contains information about the results of coprocessor operations (Figure 17). It contains dedicated bit fields which represent the exception status, condition codes, stack fault, and Top of Stack pointer. The state of the condition codes, exception status bits, and stack fault bit after execution of an instruction is provided in the description of each instruction.

Figure 17. 16-bit Status Register



The information in the Status Register can be transferred to the host system for its own processing.

- bits:* 15 B Busy Bit. This bit exists to maintain backward compatability with the 8087. It has the same value as the contents of the ES bit (bit 7). Bit B defaults to a logic 0 after a FINIT instruction. After a RESET sequence, this bit defaults to a logic 1 to indicate to the host system that the SuperMath coprocessor is present.
- 14 C3 Condition Code Bit 3. This bit, in conjunction with condition code bits C2:C0 (bits 10-8), reflects the condition of the result of an instruction's execution. Bits C3:C0 are updated by an instruction at the completion of its execution. Bit C3 defaults to a logic 0 after a RESET sequence or FINIT instruction.

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002619 TT5 ■ CHP

13:11 TS2

Top of Stack Bit Field. The value in these bits designates the Top of Stack Data Register. These bits default to logic 0 values after a RESET sequence or FINIT instruction.

TS2	TS1	TS0	Description
0	0	0	Top of Stack = Data Register 0 (default after RESET/FINIT)
0	0	1	Top of Stack = Data Register 1
0	1	0	Top of Stack = Data Register 2
0	1	1	Top of Stack = Data Register 3
1	0	0	Top of Stack = Data Register 4
1	0	1	Top of Stack = Data Register 5
1	1	0	Top of Stack = Data Register 6
1	1	1	Top of Stack = Data Register 7

10:8 C2:C0

Condition Code Bits 2-0. These bits, in conjunction with condition code bit 3 (bit 14), reflect the condition of the result of an instruction's execution. These bits are updated by an instruction at the completion of its execution. They default to logic 0 values after a RESET sequence or a FINIT instruction.

7 ES

Exception Status Bit. This bit is set if any unmasked Exception Flags (bits 5-0) are set. When this bit is set, the ERROR* output goes to an active (logic low) level. Bit ES defaults to a logic 0 after a FINIT instruction. After a RESET sequence, this bit defaults to a logic 1 to indicate to the host system that the SuperMathSX coprocessor is present.

6 S

Stack Fault Bit. This bit is set if an invalid operation occurs due to a stack underflow or overflow. When this bit is set, condition code bit 1 (C1) shows whether the fault was an underflow (C1 = 0) or an overflow (C1 = 1). Bit S defaults to a logic 0 after a RESET sequence or FINIT instruction.

5 P

Precision Exception Flag. This bit is set if a precision exception is realized as a result of an instruction's execution. A detailed explanation is provided in the "Exceptions" section of this document. Bit P defaults to a logic 0 after a RESET sequence or FINIT instruction.

4 U

Data Underflow Exception Flag. Bit U is set if a data underflow exception is realized as a result of an instruction's execution. A detailed explanation is provided in the "Exceptions" section of this document. This bit defaults to a logic 0 after a RESET sequence or FINIT instruction.

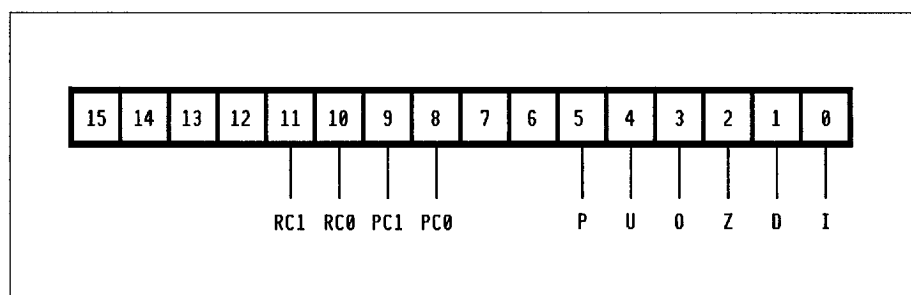
CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002620 717 ■ CHP

3	O	Data Overflow Exception Flag. Bit O is set if a data overflow exception is realized as a result of an instruction's execution. A detailed explanation is provided in the "Exceptions" section of this document. Bit O defaults to a logic 0 after a RESET sequence or FINIT instruction.
2	Z	Divide By Zero Exception Flag. Bit Z is set if an attempt to divide by zero is realized as a result of an instruction's execution. A detailed explanation is provided in the "Exceptions" section of this document. Note that this bit may be set by instructions other than FDIV-type instructions. Bit Z defaults to a logic 0 after a RESET sequence or FINIT instruction.
1	D	Denormalized Operand Exception Flag. Bit D is set if at least one of the operands on which an instruction is to be performed is a denormalized number. A detailed explanation is provided in the "Exceptions" section of this document. This bit defaults to a logic 0 after a RESET sequence or FINIT instruction.
0	I	Invalid Operation Exception Flag. Bit I is set if an invalid operation is attempted. A detailed explanation is provided in the "Exceptions" section of this document. This bit defaults to a logic 0 after a FINIT instruction; after a RESET sequence, it defaults to a logic 1.

Control Register

The 16-bit Control Register (Figure 18) specifies rounding control, precision control, and the exception masks. The rounding control bits determine the method of rounding the results of an operation. The precision control bits specify the realized precision of an operation's result. The exception mask bits are programmable masks for each of the six exception conditions.

Figure 18. 16-bit Control Register



bits: 15:13

Reserved. Software should ignore the state of these bits during reads of the contents of this register (reading the values transferred by a FSTSW instruction). All values written to these bits (with an FLDCW instruction) are ignored by the SuperMathSX coprocessor. These bits are undefined after a RESET sequence or FINIT instruction.

12

Reserved. At the completion of a RESET sequence and after execution of a FINIT instruction, the state of this bit is 0. Software should ignore the state of these bits during reads of the contents of this register (reading the values transferred by a FSTSW instruction). All values written to these bits (with an FLDCW instruction) are ignored by the SuperMathSX coprocessor.

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002622 59T ■ CHP

11:10 RC1:RC0 Rounding Control Bits 1-0. Per the following table, these bits specify the rounding method to be applied to any result that requires rounding after the instruction's completion:

RC1	RC0	Description
0	0	Round to nearest even (default)
0	1	Round toward $-\infty$
1	0	Round toward $+\infty$
1	1	Truncate toward zero

These bits default to logic 0 values after a RESET sequence or FINIT instruction.

9:8 PC1:PC0 Precision Control Bits 1-0. Per the table, these bits specify the significand's precision after the completion of FADD, FSUB, FMUL, FDIV, and FSQRT instructions.

PC1	PC0	Description
0	0	Single Precision/24-bit
0	1	Reserved
1	0	Double Precision/53-bit
1	1	Double Extended Precision/64-bit (default)

These bits default to logic 1 values after a RESET sequence or FINIT instruction.

7:6 Reserved. Software should ignore the state of these bits during reads of the contents of this register (reading the values transferred by a FSTSW instruction). All values written to these bits (with an FLDCW instruction) are ignored by the SuperMathSX coprocessor. These bits are undefined after a RESET sequence or FINIT instruction.

5 P Precision Exception Mask. Bit P masks a Precision exception should one be realized. This bit defaults to a logic 1 after a RESET sequence or FINIT instruction.

4 U Data Underflow Exception Mask. Bit U masks a Data Underflow exception should one be realized. This bit defaults to a logic 1 after a RESET sequence or FINIT instruction.

3 O Data Overflow Exception Mask. Bit O masks a Data Overflow exception should one be realized. This bit defaults to a logic 1 after a RESET sequence or FINIT instruction.

2	Z	Divide By Zero Exception Mask. Bit Z masks a Divide by Zero exception should one be realized. This bit defaults to a logic 1 after a RESET sequence or FINIT instruction.
1	D	Denormalized Operand Exception Mask. Bit D masks a Denormalized Operand exception should one be realized. This bit defaults to a logic 1 after a RESET sequence or FINIT instruction.
0	I	Invalid Operation Exception Mask. Bit I masks an Invalid Operation exception should one be realized. This bit defaults to a logic 1 after a FINIT instruction. After a RESET sequence, this bit defaults to a logic 0.

Data Register Errors

The SuperMathSX coprocessor detects the following data register errors:

- Source Register Empty
- Destination Register Full.

Source Register Empty (stack underflow) errors occur when the SuperMathSX coprocessor attempts to perform an operation with a source operand from a data register that is empty (indicated by a binary value of 11 in the data register's tag field). Similarly, Destination Register Full (stack overflow) errors occur when the SuperMathSX coprocessor attempts to perform an operation to a destination data register that is not empty (indicated by a binary value of 00, 01, or 10 in the data register's tag field).

In both instances, the SuperMathSX coprocessor reports the stack errors in the S bit field of the Status Register. When the S bit is set, the value of condition code C1 indicates whether the stack has overflowed (C1 = 1) or underflowed (C1 = 0). Both stack overflows and underflows cause Invalid Operation exceptions.

T-49-12-05

Exceptions

While performing operations, the SuperMathSX coprocessor constantly monitors data characteristics and reports all inconsistencies and errors. Errors or inconsistencies are classified as one of the following exception conditions:

- Precision
- Data Underflow
- Data Overflow
- Denormal Operand
- Divide by Zero
- Invalid Operation.

The Status Register contains a flag bit for each of the exception conditions.

When an unmasked exception is realized (corresponding exception mask bit is 0), the SuperMathSX coprocessor begins an error trap sequence, activating the ERROR* output to the system. If the unmasked exception is a Precision exception, Data Underflow exception, or Data Overflow exception, the exception is processed in accordance with the IEEE-754-1985 specification, and the result is stored in the destination before the error trap sequence is begun. System hardware generally invokes a firmware routine to process the exception.

When a masked exception is realized (corresponding exception mask bit is set), the SuperMathSX coprocessor proceeds in a prioritized predetermined manner, which is presented in Table 8. The exception handling sequence shown in Table 9 conforms to specification IEEE-754-1985. (Note that a priority of 1 has the highest ranking.)

T-49-12-05

Table 9. *SuperMathSX Coprocessor Masked Exception Handling*

Priority	Exception	Cause	SuperMathSX FPU Action With Exception Masked
1	Invalid Operation	Operation attempted on a SNaN, unsupported format, indeterminate form, stack underflow, or stack overflow. (Stack underflows and overflows also set the S bit field.)	Result is either a QNaN, indefinite integer, or indefinite BCD.
2	Divide by Zero	At some time during the instruction's execution with a non-infinite, nonzero dividend, the corresponding divisor was 0.	Result is ∞ .
3	Denormalized	At least one of the instruction's operands is denormalized.	Normal processing occurs.
4	Data Overflow	The calculated result exceeds the largest value permitted by the specified format.	Result is either the largest permitted finite value or ∞ .
5	Data Underflow	The calculated result is nonzero, but is smaller than the smallest value permitted by the specified format.	Result is either zero or denormalized.
6	Precision	The calculated result is not discretely representable in the specified format; hence, it has been rounded according to the rounding mode specified by the RC1:0 bits.	Normal processing occurs.

Precision Exception

A precision exception occurs when the result of an operation cannot be exactly (discretely) represented in the characteristic format of the destination. As an example, transcendental operations, such as FCOS, generally produce irrational results. When real repeating results, irrational results, or results more accurate than the specified destination format occur, the result is rounded according to the method specified by rounding control bits 1-0 of the Status Register. The rounded results are then presented to their destination.

Data Underflow Exception

A Data Underflow exception occurs when any nonzero result of an operation is too small to be represented by the destination's format.

With the Data Underflow exception masked, the Data Underflow exception flag is set only if the resulting 0 or denormalized number loses its precision. When this occurs, the Precision exception flag is also set.

T-49-12-05

When the Data Underflow exception is unmasked, a Data Underflow exception occurs for each resultant data underflow. The SuperMathSX coprocessor then processes the underflow in the exact fashion mandated by the IEEE-754-1985 specification.

If the instruction's destination is located in system memory, the processing halts and no result is provided. If the destination is a data register and the instruction was not an FSCALE calculation resulting in an extreme data underflow, the result is multiplied by 24576, rounded according to the manner specified by rounding control bits 1-0 of the Status Register, and stored in the destination register. If the destination is a data register and the instruction was an FSCALE calculation resulting in an extreme data underflow, a $+0/-0$ is produced, just as if the Data Underflow exception were masked.

Data Overflow Exception

A Data Overflow exception occurs when the result of an operation is too large to be represented in the specified destination. When the Data Overflow exception is masked, the exception flag is set and the result is rounded in the manner specified by the rounding control bits.

When the Data Overflow exception is unmasked, a Data Overflow exception occurs for each resultant data overflow. The SuperMathSX coprocessor then processes the overflow in the exact fashion mandated by the IEEE-754-1985 specification.

If the instruction's destination is located in system memory, the processing halts and no result is provided. If the destination is a data register and the instruction was not an FSCALE calculation resulting in an extreme data overflow, the result is divided by 24576, rounded according to the manner specified by Rounding Control bits 1-0 of the Status Register, and stored in the destination register. If the destination is a data register and the instruction was an FSCALE calculation resulting in an extreme data overflow, a $+\infty/-\infty$ is produced, just as if the Data Underflow exception were masked.

Denormal Operand Exception

T-49-12-05

A Denormal Operand exception occurs if at least one of an instruction's operands is a denormalized number. If the Denormal Operation exception is masked, the operation completes in the fashion specified by the IEEE-754-1985 specification, using the denormal number, and sets the Denormal Operand exception flag. If the Denormal Operation exception is unmasked, the Denormal Operation exception flag is set and an error trap sequence begins with no result presented to the destination.

Divide by Zero Exception

A Divide by Zero exception occurs when an instruction attempts to divide a finite nonzero operand by zero. The FDIV, FYL2X and FXTRACT instructions can all cause this exception. If the Divide by Zero exception is masked, the FDIV and FYL2X instructions return a signed infinity (with the sign resulting in the Exclusive OR of the operands) to the destination. The FXTRACT instruction places a signed zero (with the same sign as the original operand) in the Top of Stack ST(0) and a negative infinity in ST(1). If the Divide by Zero exception is unmasked, the Divide by Zero exception flag is set and an error trap sequence begins with no result presented to the destination.

Invalid Operation

Invalid Operation exceptions occur when operations that produce meaningless results are attempted. The IEEE-754-1985 document specifies the manner of detecting and reporting instances when instructions are performed on invalid operands. Invalid Operation exceptions result from both data register errors (stack underflows and overflows) and invalid operands specified in arithmetic operations.

When the Invalid Operation exception is masked, the SuperMathSX coprocessor operates as shown in Table 10, in accordance with specification IEEE-754-1985. If the Invalid Operation exception is unmasked, the Invalid Operation exception flag is set, the instruction's execution is halted, and an error trap sequence begins. No result is presented to the destination.

CHIPS & TECHNOLOGIES INC SLE D ■ 2098116 0002628 T08 ■ CHP

Table 10. Responses to Invalid Operations

T-49-12-05

Operation	Operands	Masked Response
Arithmetic	Unsupported operands	QNaN indefinite
Arithmetic	Signaling NaN	QNaN
Compare and test	One or both operands is a NaN	Set condition codes to unordered.
Addition	Opposite-signed infinities	QNaN indefinite
Subtraction	Like-signed infinities	QNaN indefinite
Multiplication	0 and ∞ operands	QNaN indefinite
Division	∞ , ∞ operands or 0, 0 operands	QNaN indefinite
Remainder: FPREM, FPREM1	Divisor = 0 or dividend = ∞	QNaN indefinite
Trigonometric: FCOS, FPTAN, FSIN, FSINCOS	Argument = ∞	QNaN indefinite
Squart root, log: FSQRT, FYL2X	Negative operand	QNaN indefinite
Integer, store: FIST(P)	Empty register, NaN, ∞	Store integer
Integer, BCD store: FBSTP	Empty register, NaN, ∞ ; or exceeds integer range	Store packed decimal indefinite
Register exchange: FXCH	Empty register	Set empty register to QNaN indefinite, then exchange.

Instruction Set

T-49-12-05

This section describes the complete set of floating-point instructions available on the SuperMathSX floating-point processor. Note that the SuperMathSX FPU instruction set is identical to that of the 80387SX.

Notation

The following sections describe how each instruction is presented. Tables in the sections define symbols and their meanings as used in the descriptions of the instructions.

Each instruction includes the following information:

- Instruction name
- Instruction syntax
- Instruction format
- Instruction description
- Condition codes
- Zero/infinity
- Exception flags.

Instruction Name

The name of an instruction (or related sets of instructions) is listed at the top of a page and is followed by a brief description of the instruction.

Table 11 defines the instruction mnemonics. Within the table, an asterisk (*) represents any number of missing characters.

Table 11. *Instruction Mnemonics*

T-49-12-05

Mnemonic	Meaning
F*	All SuperMath FPU instructions begin with F.
FI*	Instructions that operate on integer data types
FB*	Instructions that operate with BCD data types
F*P	Instructions that cause the stack to be popped once
F*PP	Instructions that cause the stack to be popped twice
F*R	The reverse form of the instruction

Syntax and Format

The Syntax shows the usage of an instruction. The Format section lists all operand sources and destinations. All arguments that must be supplied are shown in italics. Optional arguments are surrounded by square brackets. Table 12 defines the argument symbols used in the Syntax and Format sections.

Table 12. *Argument Symbols*

Symbol	Meaning
<i>dest</i>	Destination operand
<i>source</i>	Source operand
<i>stack</i>	Top of stack register represented as ST(0)
<i>register</i>	Data register represented as ST(<i>n</i>) where <i>n</i> = 0 to 7
<i>memory</i>	Memory operand address in the 80386SX

Description

The Description begins with a table showing the instruction, source operand, opcode, and clock cycles. The clock cycles are specified in terms of CPU clock cycles. The opcode is specified for each format of the instruction. Table 13 lists the types of opcodes and their location in memory.

Table 13. *Opcode Locations*

T-49-12-05

Type	Location
ST(0)	Top of Stack
ST(1)	Next to Top of Stack
ST(n)	80-bit data register, where n = 2-7 (referenced from the Top of Stack)
mem16i	Pointer to 16-bit integer in system memory
mem32i	Pointer to 32-bit integer in system memory
mem64i	Pointer to 64-bit integer in system memory
mem80b	Pointer to 80-bit BCD in system memory
mem32r	Pointer to 32-bit real number in system memory
mem64r	Pointer to 64-bit real number in system memory
mem80r	Pointer to 80-bit real number in system memory
mem2byte	Pointer to a 2-byte field in system memory
mem14/28byte	Pointer to either a 14-byte or 28-byte field in system memory
mem94/108byte	Pointer to either a 94-byte or 108-byte field in system memory
80386SX AX Reg	The AX register in an 80386SX compatible CPU
const	The specified 80-bit real number constant in the SuperMathSX coprocessor

The opcode includes the field abbreviations described in Table 14.

Table 14. *Field Abbreviations*

Abbreviation	Meaning
xxh	Hexadecimal value
SIB	Stack Index Base (from the 80386SX-compatible CPU)
displ	Displacement: an 8-bit or 32-bit value following the instruction (from the 80386SX-compatible CPU)
REG2:0	SuperMathSX FPU data register; 3 bits
RM2:0	Register/Memory bits on the 80386SX-compatible CPU; 3 bits
MD1:0	Mod bit field on the 80386SX-compatible CPU; 2 bits

The Condition Codes section shows how executing the instruction affects the status of the SuperMathSX coprocessor condition codes (C0, C1, C2, and C3).

Zero/Infinity

This section includes a table that provides information about the results produced when an instruction is executed with large operands. This section is omitted if it does not apply to an instruction. Table 15 describes the symbols used in the zero/infinity tables.

Table 15. *Zero/Infinity Symbols*

Symbol	Meaning
x,y	Positive integers
NaN	Not a number
R(0)	Zero as produced by current RC mode

T-49-12-05

Exception Flags

Table 16 lists the classes of numeric exception conditions that are recognized by the SuperMathSX coprocessor while executing numeric instructions.

Table 16. Exception Flag Symbols

Symbol	Meaning
QNaN	Quiet NaN; has 1 as the most significant bit of the significand.
Exception status:	S Stack fault or invalid register operation
	P Precision error
	U Underflow error
	O Overflow error
	Z Divide by zero error
	D Denormalized operand error
	I Invalid operation
	Blank means unaffected due to execution of instruction.
MEM	The value is loaded from memory.

A table within the Exception Flags section describes the flags that are set for the exceptions that can occur for the current instruction. Otherwise, the section indicates “no exceptions.”

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002634 201 ■ CHP

F2XM1 — Compute (2^x-1)

T-49-12-05

Syntax: **F2XM1**

Format: **F2XM1**

Description:

Instruction	Operand(s) (dest, source)	1st Byte	Opcode/Instruction Encoding								Clock Cycles	
			2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
F2XM1	ST(0)	D9h	1	1	1	1	0	0	0	0	8-141	

F2XM1 replaces the value at the top of the stack with 2^x-1 , where x is the original value in the Top of Stack (where $-1 \leq x \leq 1$). The result is rounded, depending upon the mode in effect.

Condition Codes: C0, C2, and C3 are undefined. C1 is set to 0 after a Precision exception, where it defines whether or not rounding is away from 0.

Zero/Infinity:

Operand	Result	Operand	Result
+0	+0	$-\infty$	-1
-0	-0	$+\infty$	$+\infty$

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002635 148 ■ CHP

Exception Flags:

T-49-12-05

Exception	Mode	Result	S	P	U	O	Z	D	I
Register Error	Masked	QNaN	1						1
	Unmasked	Trap, abort	1						1
Precision	Masked	Rounded		1					
	Unmasked	Rounded		1					
Underflow	Masked	Denorm,0		1	1				
	Unmasked	Round, scale			1				
Denormal	Masked	Operand used						1	
	Unmasked	Trap, unchanged						1	
Invalid Operation	Masked	QNaN							1
	Unmasked	Trap, unchanged							1

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002636 084 ■ CHP

FABS — Absolute value

T-49-12-05

Syntax: **FABS**Format: **FABS**

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FABS	ST(0)	D9h	1	1	1	0	0	0	0	1	4	

The FABS instruction replaces the value at the top of the stack with its absolute value. FABS sets the sign of the Top of Stack to 0 (i.e., positive).

Condition Codes: C0, C2, and C3 are undefined. C1 is set to 0.

Zero/Infinity:

Operand	Result	Operand	Result
+0	+0	$-\infty$	$+\infty$
-0	+0	$+\infty$	$+\infty$

Exception Flags:

Exception	Mode	Result	S	P	U	O	Z	D	I
Register	Masked	QNaN	1						1
Error	Unmasked	Trap, Abort	1						1

CHIPS & TECHNOLOGIES INC SLE D ■ 2098116 0002637 T10 ■ CHP

FADD — Floating-point add

T-49-12-05

Syntax: **FADD(P) [[*dest*,]*source*]**
FIADD [[*dest*,]*source*]

Format: **FADD *stack, memory***
FADD *stack, register*
FADD(P) *register, stack*
FIADD *stack, memory*

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FIADD	ST(0),mem16i	DEh	MD1	MD0	0	0	0	RM2	RM1	RM0	SIB,displ	12
FIADD	ST(0),mem32i	DAh	MD1	MD0	0	0	0	RM2	RM1	RM0	SIB,displ	12
FADD	ST(0),mem32r	D8h	MD1	MD0	0	0	0	RM2	RM1	RM0	SIB,displ	10
FADD	ST(0),mem64r	DCh	MD1	MD0	0	0	0	RM2	RM1	RM0	SIB,displ	10
FADD	ST(0),ST(n)	D8h	1	1	0	0	0	REG2	REG1	REG0		7
FADD	ST(n),ST(0)	DCh	1	1	0	0	0	REG2	REG1	REG0		7
FADDP	ST(n),ST0	DEh	1	1	0	0	0	REG2	REG1	REG0		7

The FADD instruction fetches, then adds the operands. The result is stored in the destination. The source operand is changed to extended precision format, if required. Depending on the mode in effect, the result is rounded to the precision indicated by the mode bits. FADDP causes the stack to be popped once.

Condition Codes: C0, C2, and C3 are undefined. C1 is set to 0 except after a Precision exception, where it defines whether or not rounding is away from 0.

Zero/Infinity:

T-49-12-05

Operand 1	Operand 2	Result	Operand 1	Operand 2	Result
+0	+0	+0	+∞	+∞	+∞
-0	-0	-0	-∞	-∞	-∞
+0	-0	R(0)	+∞	-∞	Invalid
-0	+0	R(0)	-∞	+∞	Invalid
-x	+x	R(0)	+∞	x	+∞
+x	-x	R(0)	-∞	x	-∞

Exception Flags:

Exception	Mode	Result	S	P	U	O	Z	D	I
Register	Masked	QNaN	1						1
Error	Unmasked	Unchanged	1						1
Precision	Masked	Rounded		1					
	Unmasked	Rounded		1					
Underflow	Masked	Denormal/0		1	1				
	Unmasked	Round, scale			1				
Overflow	Masked	Infinity				1			
	Unmasked	Round, scale				1			
Denormal	Masked	Denorm						1	
	Unmasked	Trap, abort						1	
Invalid	Masked	QNaN							1
Operation	Unmasked	Unchanged							1

FCHS — Change sign

T-49-12-05

Syntax: FCHS

Format: FCHS

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FCHS	ST(0)	D9h	1	1	1	0	0	0	0	0	4	

The FCHS instruction replaces the value at the Top of Stack with its opposite.

Condition Codes: C0, C2, and C3 are undefined. C1 is set to 0.

Zero/Infinity:

Operand	Result	Operand	Result
+0	-0	+∞	-∞
-0	+0	-∞	+∞

Exception Flags:

Exception	Mode	Result	S	P	U	O	Z	D	I
Register	Masked	QNaN	1						1
Error	Unmasked	Trap, Abort	1						1

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002640 505 ■ CHP

FCLEX— Clear exceptions

T-49-12-05

Syntax: **FCLEX**

Format: **FCLEX**

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FCLEX	ST(n),ST(0)	DBh	1	1	1	0	0	0	1	0	4	

The FCLEX instruction resets all exception flags, the Exception Status flag, and the Busy flag to 0.

Condition Codes: C0, C1, C2, and C3 are undefined.

Exception Flags:

Exception	Result	S	P	U	O	Z	D	I
All	Reset to zero	0	0	0	0	0	0	0

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002641 441 ■ CHP

FCOM— Floating-point compare

T-49-12-05

Syntax: **FCOM(P) (P) [[*dest*,]*source*]**

Format:

FICOM(P) *stack, memory*
FCOM(P) *stack, memory*
FCOM *stack, register*
FCOMP *stack, register*
FCOM (P)(P) *stack, register*

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FICOM	ST(0),mem16i	DEh	MD1	MD0	0	1	0	RM2	RM1	RM0	SIB,displ	13
FICOMP	ST(0),mem16i	DEh	MD1	MD0	0	1	1	RM2	RM1	RM0	SIB,displ	13
FICOM	ST(0),mem32i	DAh	MD1	MD0	0	1	0	RM2	RM1	RM0	SIB,displ	10
FICOMP	ST(0),mem32i	DAh	MD1	MD0	0	1	1	RM2	RM1	RM0	SIB,displ	13
FCOM	ST(0),mem32r	D8h	MD1	MD0	0	1	0	RM2	RM1	RM0	SIB,displ	10
FCOMP	ST(0),mem32r	D8h	MD1	MD0	0	1	1	RM2	RM1	RM0	SIB,displ	10
FCOM	ST(0),mem64r	DCh	MD1	MD0	0	1	0	RM2	RM1	RM0	SIB,displ	12
FCOMP	ST(0),mem64r	DCh	MD1	MD0	0	1	1	RM2	RM1	RM0	SIB,displ	12
FCOM	ST(n),ST(0)	D8h	1	1	0	1	0	REG2	REG1	REG0		8
FCOMP	ST(n),ST(0)	D8h	1	1	0	1	1	REG2	REG1	REG0		5
FCOMPP	ST(1),ST(0)	DEh	1	1	0	1	1	0	0	1		8

The Compare instructions compare the top of the stack to the source operand. The source operand can be a register, a single or double real, or a word or long integer, and it is converted to extended precision, if necessary. Exception flags are used to show the result of the comparison.

If either operand is a NaN or is an unsupported number, or if a stack fault occurs, the Invalid Operation exception flag is set and the result is set to Unordered. If the operands are QNaNs, the result is an invalid operation.

CHIPS & TECHNOLOGIES INC

51E D ■ 2098116 0002642 388 ■ CHP

Condition Codes:

The result of the comparison is determined by the condition code bits C3, C2, C1, and C0, as follows:

T-49-12-05

dest/source Status	C3	C2	C1	C0
dest > source	0	0	0	0
dest < source	0	0	0	1
dest = source	1	0	0	0
Unordered	1	1	0	1

If the source operand is either NaN or an undefined value, or if a stack fault occurs, an Invalid Operation exception occurs and the condition bits are undefined.

If the source register is empty, C3 = C2 = C0 = 1; C1 = 0.

Zero/Infinity:

Destination	Source	Result	Destination	Source	Result
+0	+0	equal	+∞	+∞	equal
-0	-0	equal	-∞	-∞	equal
+0	-0	equal	+∞	-∞	dest>source
-0	+0	equal	-∞	+∞	dest<source
+0	+x	dest<source	+∞	x	dest>source
-0	+x	dest<source	-∞	x	dest<source
+0	-x	dest>source	x	-∞	dest>source
-0	-x	dest>source	x	+∞	dest<source
NaN	x	Unordered	x	NaN	Unordered

Exception Flags:

Exception	Mode	Result	S	P	U	O	Z	D	I
Register Error	Masked	Unordered	1						1
	Unmasked	Trap, abort	1						1
Denormal	Masked	QNaN						1	
	Unmasked	Trap, abort						1	
Invalid Operation	Masked	Unordered							1
	Unmasked	Trap, abort							1

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002643 214 ■ CHP

FCOS — Compute cosine(x)

T-49-12-05

Syntax: **FCOS**Format: **FCOS**

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FCOS	ST(0)	D9h	1	1	1	1	1	1	1	1	5-129	

The FCOS instruction performs the function evaluation: $y = \cos(x)$. The source operand, x , is taken from the Top of Stack and must be in the range $0 < |x| < 2^{63}$. The result, y , is rounded according to the mode in effect and returned to the Top of Stack.

Condition Codes: C0 and C3 are undefined. C2 is used to specify reduction; it is set to 1 and is incomplete if the operand at the top of the stack is out of range. C1 is set to 1 after a Precision exception if the rounding done by the instruction was upward.

Zero/Infinity:

Operand	Result	Operand	Result
+0	+1	$+\infty$	Invalid operation
-0	+1	$-\infty$	Invalid operation

CHIPS & TECHNOLOGIES INC SLE D ■ 2098116 0002644 150 ■ CHP

Exception Flags:

T-49-12-05

Exception	Mode	Result	S	P	U	O	Z	D	I
Register Error	Masked	QNaN	1						1
	Unmasked	Unchanged	1						1
Precision	Masked	Rounded		1					
	Unmasked	Rounded		1					
Denormal	Masked	Operand used						1	
	Unmasked	Trap, unchanged						1	
Invalid Operation	Masked	QNaN							1
	Unmasked	Unchanged							1

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002645 097 ■ CHP

FDECSTP — Decrement stack pointer

T-49-12-05

Syntax: **FDECSTP**Format: **FDECSTP**

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FDECSTP	None	D9h	1	1	1	1	0	1	1	0		4

The FDECSTP instruction subtracts one (modulo 8) from the data register Top of Stack pointer (TS2:0 of the Status Register).

Condition Codes: C0, C2, and C3 are undefined. C1 is set to 0.

Exception Flags: No exceptions.

FDIV — Floating-point divide

T-49-12-05

Syntax: **FDIV(R)(P) [[dest,],source]**

Format: **FDIV(R) stack, memory**
FDIV(R) stack, memory
FDIV(R) stack, register
FDIV(R)(P) register, stack

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding										Clock Cycles
		1st Byte	2nd Byte								Bytes 3-7	
			B7	B6	B5	B4	B3	B2	B1	B0		
FIDIV	ST(0),mem16i	DEh	MD1	MD0	1	1	0	RM2	RM1	RM0	SIB,displ	29-51
FIDIVR	ST(0),mem16i	DEh	MD1	MD0	1	1	1	RM2	RM1	RM0	SIB,displ	29-51
FIDIV	ST(0),mem32i	DAh	MD1	MD0	1	1	0	RM2	RM1	RM0	SIB,displ	29-51
FIDIVR	ST(0),mem32i	DAh	MD1	MD0	1	1	1	RM2	RM1	RM0	SIB,displ	29-51
FDIV	ST(0),mem32r	D8h	MD1	MD0	1	1	0	RM2	RM1	RM0	SIB,displ	28-48
FDIVR	ST(0),mem32r	D8h	MD1	MD0	1	1	1	RM2	RM1	RM0	SIB,displ	28-48
FDIV	ST(0),mem64r	DCh	MD1	MD0	1	1	0	RM2	RM1	RM0	SIB,displ	29-50
FDIVR	ST(0),mem64r	DCh	MD1	MD0	1	1	1	RM2	RM1	RM0	SIB,displ	29-50
FDIV	ST(0),ST(n)	D8h	1	1	1	1	0	REG2	REG1	REG0		24-45
FDIVR	ST(0),ST(0)	D8h	1	1	1	1	1	REG2	REG1	REG0		24-45
FDIV	ST(n),ST(0)	DCh	1	1	1	1	0	REG2	REG1	REG0		24-45
FDIVR	ST(n),ST(0)	DCh	1	1	1	1	1	REG2	REG1	REG0		24-45
FDIVP	ST(n),ST(0)	DEh	1	1	1	1	0	REG2	REG1	REG0		24-45
FDIVRP	ST(n),ST(0)	DEh	1	1	1	1	1	REG2	REG1	REG0		24-45

The FDIV instructions divide the Top of Stack by the other operand. The quotient is rounded to the precision according to the mode in effect and then placed in the destination.

Divide instructions ending in R (reverse) divide the Top of Stack into the other operand. These instructions each contain one additional clock cycle.

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002647 96T ■ CHP

Condition Codes: C0, C2, and C3 are undefined. C1 is set to 0 except after a Precision exception, where it defines whether or not rounding is away from 0.

T-49-12-05

Zero/Infinity:

Operand 1	Operand 2	Result	Operand 1	Operand 2	Result
+0	+0	Invalid	$+\infty$	$+\infty$	Invalid
+0	-0	Invalid	$+\infty$	$+\infty$	Invalid
-0	+0	Invalid	$-\infty$	$+\infty$	Invalid
-0	-0	Invalid	$-\infty$	$-\infty$	Invalid
+0	+x	+0	$+\infty$	+x	$+\infty$
-0	-x	+0	$+\infty$	-x	$-\infty$
+0	-x	-0	$-\infty$	+x	$-\infty$
-0	+x	-0	$-\infty$	-x	$+\infty$
+x	+y	$+0^*$	$+\infty$	+0	$+\infty$
-x	-y	$+0^*$	$+\infty$	-0	$-\infty$
+x	-y	-0^*	$-\infty$	+0	$-\infty$
-x	+y	-0^*	$-\infty$	-0	$+\infty$

* These cases apply when $\left(\frac{+x}{+y}\right)$ produces an extreme denormalization or underflow (when the Underflow exception is masked).

Note: $\frac{\text{Dividend}}{\text{Divisor}} = \text{Result}$

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002648 8T6 ■ CHP

Exception Flags:

T-49-12-05

Exception	Mode	Result	S	P	U	O	Z	D	I
Register Error	Masked	QNaN	1						1
	Unmasked	Unchanged	1						1
Precision	Masked	Rounded		1					
	Unmasked	Rounded		1					
Underflow	Masked	Denormal/0		1	1				
	Unmasked	Round, scale			1				
Overflow	Masked	Infinity				1			
	Unmasked	Round, scale				1			
Divide by Zero	Masked	Infinity*					1		
	Unmasked	Trap, unchanged					1		
Denormal	Masked	Operand used						1	
	Unmasked	Trap, unchanged						1	
Invalid Operation	Masked	QNaN							1
	Unmasked	Trap, unchanged							1

* When the Divide by Zero exception is masked, the results in the above Zero/Infinity table are produced.

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002649 732 ■ CHP

FFREE — Free floating-point register

T-49-12-05

Syntax: **FFREE** [*dest*]Format: **FFREE** *register*

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles		
		1st Byte	2nd Byte									Bytes 3-7	
			B7	B6	B5	B4	B3	B2	B1	B0			
FFREE	ST(n)	DDh	1	1	0	0	0	REG2	REG1	REG0	3		

The FFREE instruction changes the destination register tag to empty, but it does not alter the contents of the register.

Condition Codes: C0, C1, C2, and C3 are undefined.

Exception Flags: No exceptions.

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002650 454 ■ CHP

FINCSTP— Increment stack pointer

T-49-12-05

Syntax: **FINCSTP**

Format: **FINCSTP**

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FINCSTP	None	D9h	1	1	1	1	0	1	1	1	3	

The FINCSTP instruction adds one (modulo 8) to the data register Top of Stack pointer.

Condition Codes: C0, C2, and C3 are undefined. C1 is set to 0.

Exception Flags: No exceptions.

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002651 390 ■ CHP

FINIT — Initialize SuperMathSX floating-point unit

T-49-12-05

Syntax: **FINIT**Format: **FINIT**

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles
		1st Byte	2nd Byte							Bytes 3-7	
FINIT	None	DBh	1	1	1	0	0	0	1	1	3

The FINIT instruction resets the SuperMathSX floating-point coprocessor as shown in Table 17.

Table 17. FINIT Instruction Reassignments

Item	Reset Value
Mode control register	037Fh
Status register	0000h
Data registers Tag Word	FFFFh (Data Registers Empty)
Rounding control	RC1:0 = 00 (round to nearest)
Precision	PC1:0 = 11 (64-bit extended)
Top of stack	TS2:0 = 000
Exception flags	Cleared (0)
Condition codes (C0-C3)	Cleared (C0 = C1 = C2 = C3 = 0)

Condition Codes: C0, C1, C2, and C3 are set to 0.

Exception Flags: No exceptions.

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002652 227 ■ CHP

FLD— Load Top of Stack

T-49-12-05

Syntax: **FLD** *[[dest,]source]*

Format: **FLD** *stack, memory*
FLD *stack, register*

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FILD	ST(0),mem16i	DFh	MD1	MD0	0	0	0	RM2	RM1	RM0	SIB,displ	4
FILD	ST(0),mem32i	DBh	MD1	MD0	0	0	0	RM2	RM1	RM0	SIB,displ	4
FILD	ST(0),mem64i	DFh	MD1	MD0	1	0	1	RM2	RM1	RM0	SIB,displ	4
FBLD	ST(0),mem80b	DFh	MD1	MD0	1	0	0	RM2	RM1	RM0	SIB,displ	63
FLD	ST(0),mem32r	D9h	MD1	MD0	0	0	0	RM2	RM1	RM0	SIB,displ	3
FLD	ST(0),mem64r	DDh	MD1	MD0	0	0	0	RM2	RM1	RM0	SIB,displ	3
FLD	ST(0),mem80r	DBh	MD1	MD0	1	0	1	RM2	RM1	RM0	SIB,displ	2
FLD	ST(0),ST(n)	D9h	1	1	0	0	0	REG2	REG1	REG0		3

The FLD instruction fetches the source operand. If the source operand is in single or double precision format, word or long integer format, or packed decimal format, the FLD instruction converts it into extended precision format. The result is stored in the TS2:0 bits field of the Status Word.

The Top of Stack is decremented.

ST7 must be empty or else an invalid operation exception will occur. Precision, Underflow, Overflow, and Divide by Zero exceptions cannot occur as a result of FLD. Denormal Operation exceptions cannot occur as a result of 80-bit real or FLD register instructions.

Condition Codes: C0, C2, and C3 are undefined. C1 is set to 0 after normal execution. In the case of a register error, C1 is set to 1 if the destination register is full and to 0 if the source register is empty.

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002653 163 ■ CHP

Zero/Infinity:

T-49-12-05

Operand	Result	Operand	Result
+0	+0	$+\infty$	$+\infty$
-0	-0	$-\infty$	$-\infty$

Exception Flags:

Exception	Mode	Result	S	P	U	O	Z	D	I
Register Error	Masked	QNaN	1						1
	Unmasked	Unchanged	1						1
Denormal	Masked	Denorm						1	
	Unmasked	Trap, abort						1	
Invalid Operation	Masked	QNaN							1
	Unmasked	Unchanged							1

FLD1 — Load constant 1.0

T-49-12-05

Syntax: **FLD1**

Format: **FLD1**

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FLD1	ST(0),const	D9h	1	1	1	0	1	0	0	0	2	

FLD1 pushes 1.0 onto the stack.

Condition Codes: C0, C2, and C3 are undefined. C1 is 0 after normal execution. C1 is set to 1 after a register error, indicating that the destination register is full.

Exception Flags:

Exception	Mode	Result	S	P	U	O	Z	D	I
Register	Masked	QNaN	1						1
Error	Unmasked	Unchanged	1						1

T-49-12-05

FLDCW — Load mode control word

Syntax: **FLDCW** *source*

Format: **FLDCW** *memory*

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FLDCW	mem2byte	D9h	MD1	MD0	1	0	1	RM2	RM1	RM0	5	

The FLDCW instruction loads the specified bytes into the Mode Control Word and is typically used to establish the mode of operation.

Condition Codes: C0, C1, C2, and C3 are undefined.

Exception Flags: No exceptions.

T-49-12-05

FLDENV — Load SuperMathSX coprocessor environment

Syntax: **FLDENV** *source*

Format: **FLDENV** *memory*

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FLDENV	mem14/28byte	D9h	MD1	MD0	1	0	0	RM2	RM1	RM0	92	

The FLDENV instruction loads the SuperMathSX FPU environment from the specified memory location. The mode of the 80386SX-compatible CPU and the operand size determine the format of the environment as shown in Figure 19. The environment is made up of the Mode Control Word, the Status Word, the Data Register Tag Word, the Instruction Pointer, and the Data Pointer.

Condition Codes: C0, C1, C2, and C3 are loaded from memory.

Exception Flags: Exceptions may be loaded with the environment. If the Status Word loaded with the environment contains an exception that is enabled, an error trap will occur after the next wait or after the execution of an Exception Status instruction.

T-49-12-05

Figure 19. SuperMathSX Coprocessor Environment

16-Bit Protected Mode

Byte+1								Byte+0								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Mode Control Word																00h
Status Word																02h
Tag Word																04h
Instruction Pointer Offset																06h
Code Segment Selector																08h
Operand Offset																0Ah
Operand Segment Selector																0Ch

16-Bit Real Mode/Virtual 8086 Mode

Byte+1								Byte+0								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Mode Control Word																00h
Status Word																02h
Tag Word																04h
Instruction Pointer 15:00																06h
IP 19:16		0		Opcode 10:0												08h
Operand Pointer 15:00																0Ah
OP 19:16		0		0		0		0		0		0		0		0Bh

32-Bit Protected Mode

Byte+3								Byte+2								Byte+1								Byte+0								
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved																Mode Control Word																00h
Reserved																Status Word																04h
Reserved																Tag Word																08h
Instruction Pointer Offset																																0Ch
0	0	0	0	0	0	Opcode 10:0										Code Segment Selector																10h
Operand Offset																																14h
Reserved																Operand Segment Selector																18h

32-Bit Real Mode

Byte+3								Byte+2								Byte+1								Byte+0											
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
Reserved																Mode Control Word																00h			
Reserved																Status Word																04h			
Reserved																Tag Word																08h			
Reserved																Instruction Pointer 15:00																0Ch			
0	0	0	0	Instruction Pointer 31:16												0	Opcode 10:00												10h						
Reserved																Operand Pointer 15:00																14h			
0	0	0	0	Operand Pointer 31:16												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18h

T-49-12-05

FLDL2E — Load constant $\log_2(e)$

Syntax: **FLDL2E**

Format: **FLDL2E**

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FLDL2E	ST(0),const	D9h	1	1	1	0	1	0	1	0	2	

FLDL2E rounds the extended precision constant $\log_2(e)$ according to the RC mode in effect and pushes it onto the stack.

Condition Codes: C0, C2, and C3 are undefined. C1 is 0 after normal execution. C1 is set to 1 after a register error, indicating that the destination register is full.

Exception Flags:

Exception	Mode	Result	S	P	U	O	Z	D	I
Register	Masked	QNaN	1						1
Error	Unmasked	Trap, abort	1						1

T-49-12-05

FLDL2T — Load constant $\log_2(10)$

Syntax: **FLDL2T**

Format: **FLDL2T**

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FLDL2T	ST(0),const	D9h	1	1	1	0	1	0	0	1	2	

FLDL2T rounds the extended precision constant $\log_2(10)$ according to the RC mode in effect and pushes it onto the stack.

Condition Codes: C0, C2, and C3 are undefined. C1 is 0 after normal execution. C1 is set to 1 after a register error, indicating that the destination register is full.

Exception Flags:

Exception	Mode	Result	S	P	U	O	Z	D	I
Register	Masked	QNaN	1						1
Error	Unmasked	Trap, abort	1						1

FLDLG2— Load constant $\log_{10}(2)$

T-49-12-05

Syntax: **FLDLG2**

Format: **FLDLG2**

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FLDLG2	ST(0),const	D9h	1	1	1	0	1	1	0	0	2	

FLDLG2 rounds the extended precision constant $\log_{10}(2)$ according to the RC mode in effect and pushes it onto the stack.

Condition Codes: C0, C2, and C3 are undefined. C1 is 0 after normal execution. C1 is set to 1 after a register error, indicating that the destination register is full.

Exception Flags:

Exception	Mode	Result	S	P	U	O	Z	D	I
Register	Masked	QNaN	1						1
Error	Unmasked	Trap, abort	1						1

T-49-12-05

FLDLN2 — Load constant ln(2)

Syntax: **FLDLN2**

Format: **FLDLN2**

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FLDLN2	ST(0),const	D9h	1	1	1	0	1	1	0	1	2	

FLDLN2 rounds the extended precision constant ln(2) according to the RC mode in effect and pushes it onto the stack.

Condition Codes: C0, C2, and C3 are undefined. C1 is 0 after normal execution. C1 is set to 1 after a register error, indicating that the destination register is full.

Exception Flags:

Exception	Mode	Result	S	P	U	O	Z	D	I
Register	Masked	QNaN	1						1
Error	Unmasked	Trap, abort	1						1

T-49-12-05

FLDPI— Load constant π Syntax: **FLDPI**Format: **FLDPI**

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FLDPI	ST(0),const	D9h	1	1	1	0	1	0	1	1	2	

FLDPI rounds the extended precision constant that approximates pi (π) according to the RC mode in effect and pushes it onto the stack.

Condition Codes: C0, C2, and C3 are undefined. C1 is 0 after normal execution. C1 is set to 1 after a register error, indicating that the destination register is full.

Exception Flags:

Exception	Mode	Result	S	P	U	O	Z	D	I
Register Error	Masked	QNaN	1						1
	Unmasked	Trap, abort	1						1

FLDZ — Load constant 0

T-49-12-05

Syntax: **FLDZ**Format: **FLDZ**

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FLDZ	ST(0),const	D9h	1	1	1	0	1	1	1	0		4

FLDZ pushes zero onto the stack.

Condition Codes: C0, C2, and C3 are undefined. C1 is 0 after normal execution. C1 is set to 1 after a register error, indicating that the destination register is full.

Exception Flags:

Exception	Mode	Result	S	P	U	O	Z	D	I
Register	Masked	QNaN	1						1
Error	Unmasked	Trap, abort	1						1

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002664 T49 ■ CHP

FMUL — Floating-point multiply

T-49-12-05

Syntax: **FMUL(P) [[dest]source]**
FIMUL [[dest,]source]

Format: **FMUL stack, memory**
FMUL stack, register
FMUL(P) register, stack
FIMUL stack, memory
FMUL stack, register

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FIMUL	ST(0),mem16i	DEh	MD1	MD0	0	0	1	RM2	RM1	RM0	SIB,displ	12-15
FIMUL	ST(0),mem32i	DAh	MD1	MD0	0	0	1	RM2	RM1	RM0	SIB,displ	12
FMUL	ST(0),mem32r	D8h	MD1	MD0	0	0	1	RM2	RM1	RM0	SIB,displ	10-12
FMUL	ST(0),mem64r	DCh	MD1	MD0	0	0	1	RM2	RM1	RM0	SIB,displ	15
FMUL	ST(0),ST(n)	D8h	1	1	0	0	1	REG2	REG1	REG0		11
FMUL	ST(n),ST(0)	DCh	1	1	0	0	1	REG2	REG1	REG0		11
FMULP	ST(n),ST(0)	DEh	1	1	0	0	1	REG2	REG1	REG0		11

The FMUL instruction fetches, then multiplies the operands. It changes the source into extended precision format, if necessary. The mode in effect determines how the result is rounded. The result is stored in the destination register.

Condition Codes: C0, C2, and C3 are undefined. C1 is set to 1 after a Precision exception if the rounding done by the instruction was away from zero.

Zero/Infinity:

T-49-12-05

Operand 1	Operand 2	Result	Operand 1	Operand 2	Result
+0	+0	+0	+∞	+∞	+∞
+0	-0	-0	+∞	-∞	-∞
-0	+0	-0	-∞	+∞	-∞
-0	-0	+0	-∞	-∞	+∞
+x	+0	+0	+∞	+x	+∞
+x	-0	-0	+∞	-x	-∞
-x	+0	-0	-∞	+x	-∞
-x	-0	+0	-∞	-x	+∞
+x	+y	+0*	+∞	+0	Invalid
+x	-y	-0*	+∞	-0	Invalid
-x	+y	-0*	-∞	+0	Invalid
-x	-y	+0*	-∞	-0	Invalid

* For cases in which the product of xy produces an extremely small number (i.e., underflow) and the Underflow exception is masked, the result is denormalized to 0.

Exception Flags:

Exception	Mode	Result	S	P	U	O	Z	D	I
Register Error	Masked	QNaN	1						1
	Unmasked	Unchanged	1						1
Precision	Masked	Rounded		1					
	Unmasked	Rounded		1					
Underflow	Masked	Denormal/0		1	1				
	Unmasked	Round, scale		1	1				
Overflow	Masked	Infinity				1			
	Unmasked	Round, scale				1			
Denormal	Masked	Denormal used						1	
	Unmasked	Trap, abort						1	
Invalid Operation	Masked	QNaN							1
	Unmasked	Trap, unchanged							1

FNOP — No operation

T-49-12-05

Syntax: **FNOP**

Format: **FNOP**

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FNOP	None	D9h	1	1	0	1	0	0	0	0	4	

FNOP performs no operation and affects only instruction pointers.

Condition Codes: C0, C1, C2, and C3 are undefined.

Exception Flags: No exceptions.

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002667 758 ■ CHP

FPATAN — Arctangent

T-49-12-05

Syntax: **FPATAN**Format: **FPATAN**

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FPATAN	ST(0)	D9h	1	1	1	1	0	0	1	1		20-261

FPATAN computes the arctangent function evaluation: $z = \tan^{-1} \left(\frac{y}{x} \right)$, where z is in radians. The first source operand, x , is the Top of Stack, ST(0); the second source operand, y , is the next to Top of Stack, ST(1). The result is rounded according to the mode in effect. The result, z , falls into the range $-\pi \leq z \leq \pi$. The instruction pops the Top of Stack and returns z to the new Top of Stack.

Condition Codes: C0, C2, and C3 are undefined. C1 is 0 after normal execution. C1 is set to 1 after a Precision exception if the rounding done by the instruction was away from zero.

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002668 694 ■ CHP

Zero/Infinity:

T-49-12-05

y	x	Result
y=+0	$+\infty \geq x \geq +0$	z=+0
y=-0	$+\infty \geq x \geq +0$	z=-0
y=+0	$-\infty \leq x \leq -0$	z=+ π
y=-0	$-\infty \leq x \leq -0$	z=- π
y>+0	x=0	z=+ $\pi/2$
y>+0	$x=+\infty$	z=+0
y>+0	$x=-\infty$	z=+ π
y<-0	x=0	z=- $\pi/2$
y<-0	$x=+\infty$	z=-0
y<-0	$x=-\infty$	z=- π
y=+ ∞	$-\infty < x < +\infty$	z=+ $\pi/2$
y=+ ∞	$x=+\infty$	z=+ $\pi/4$
y=+ ∞	$x=-\infty$	z=+3 $\pi/4$
y=- ∞	$-\infty < x < +\infty$	z=- $\pi/2$
y=- ∞	$x=+\infty$	z=- $\pi/4$
y=- ∞	$x=-\infty$	z=-3 $\pi/4$

Exception Flags:

Exception	Mode	Result	S	P	U	O	Z	D	I
Register Error	Masked	QNaN	1						1
	Unmasked	Unchanged	1						1
Precision	Masked	Rounded		1					
	Unmasked	Rounded		1					
Underflow	Masked	Denormal/0		1	1				
	Unmasked	Round, scale			1				
Denormal	Masked	Denormal used						1	
	Unmasked	Trap, abort						1	
Invalid Operation	Masked	QNaN							1
	Unmasked	Unchanged							1

T-49-12-05

FPREM— Remainder

Syntax: **FPREM**

Format: **FPREM**

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FPREM	ST(0)	D9h	1	1	1	1	1	0	0	0	42-124	

The FPREM instruction computes the remainder resulting from dividing source operand x, by the next to Top of Stack operand, y. The remainder, r, replaces the value in the Top of Stack. If the numbers divide evenly, the remainder is 0. FPREM can reduce the exponent of x by 32 or more each pass.

The remainder is determined by multiplying y by the quotient and subtracting the result from x. The quotient is the result of dividing x by y, then using the chopping method to truncate the exact value toward zero. The sign of the remainder is the same as that of the original value of x in the stack.

FPREM is provided for compatibility with industry standard 8087 and 80287 numeric coprocessors. However, FPREM does not adhere to the remainder operation specified in IEEE Standard 754. FPREM1 is compatible with the IEEE standard.

Condition Codes: When the quotient is completely reduced, its value can be read from the condition code bits C3, C1, and C0, as shown in the following table.

Result of Instruction's Execution		C3	C2	C1	C0
Reduction not completed		0	1	0	0
Reduction completed: $q(\text{mod } 8)$	0	0	0	0	0
	1	0	0	1	0
	2	1	0	0	0
	3	1	0	1	0
	4	0	0	0	1
	5	0	0	1	1
	6	1	0	0	1
	7	1	0	1	1

If the source register is empty, then C0, C2, and C3 are undefined; C1 is 0.

Zero/Infinity:

x	y	Result
$x=0$	$y=0$	Invalid operation
$x \neq 0$	$y=0$	Invalid operation
$x=-0$	$y \neq 0$	-0
$x=+0$	$y \neq 0$	+0
$x=\infty$		Invalid operation
$-\infty < x < +\infty$	$y=\infty$	$x-q=0$

Exception Flags:

Exception	Mode	Result	S	P	U	O	Z	D	I
Register Error	Masked	QNaN	1						1
	Unmasked	Unchanged	1						1
Underflow	Masked	Denormal							
	Unmasked	Round, scale			1				
Denormal	Masked	Denormal used						1	
	Unmasked	Trap, abort						1	
Invalid Operation	Masked	QNaN							1
	Unmasked	Unchanged							1

T-49-12-05

FPREM1 — IEEE remainder

Syntax: **FPREM1**

Format: **FPREM1**

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FPREM1	ST(0)	D9h	1	1	1	1	0	1	0	1	50-128	

The FPREM1 instruction computes the remainder resulting from dividing source operand, x, by the next to ST(0) operand, y. The remainder, r, replaces the value in the Top of Stack. If the numbers divide evenly, the remainder is 0. FPREM1 can reduce the exponent of x by 32 or more on each pass.

The remainder is determined by multiplying y by the quotient and subtracting the result from x. The quotient is the result of dividing x by y, then rounding to the nearest number (or nearest even number in case of a tie).

The FPREM1 instruction is compatible with IEEE Standard 754.

Condition Codes: When the quotient is completely reduced, its value can be read from the condition code bits C3, C1, and C0, as shown in the following table.

T-49-12-05

Result of Instruction's Execution		C3	C2	C1	C0
Reduction not completed		0	1	0	0
Reduction completed: $q(\bmod 8)$	0	0	0	0	0
	1	0	0	1	0
	2	1	0	0	0
	3	1	0	1	0
	4	0	0	0	1
	5	0	0	1	1
	6	1	0	0	1
	7	1	0	1	1

If the source register is empty, then C0, C2, and C3 are undefined; C1 is 0.

Zero/Infinity:

x	y	Result
$x=0$	$y=0$	Invalid operation
$x \neq 0$	$y=0$	Invalid operation
$x=-0$	$y \neq 0$	-0
$x=+0$	$y \neq 0$	+0
$x=\infty$		Invalid operation
$-\infty < x < +\infty$	$y=\infty$	$x=q=0$

Exception Flags:

Exception	Mode	Result	S	P	U	O	Z	D	I
Register Error	Masked	QNaN	1						1
	Unmasked	Unchanged	1						1
Underflow	Masked	Denormal							
	Unmasked	Round, scale			1				
Denormal	Masked	Denormal used						1	
	Unmasked	Trap, abort						1	
Invalid Operation	Masked	QNaN							1
	Unmasked	Unchanged							1

T-49-12-05

FPTAN — TangentSyntax: **FPTAN**Format: **FPTAN**

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FPTAN	ST(0)	D9h	1	1	1	1	0	0	1	0		5-166

The FPTAN instruction calculates the tangent of x , the source operand taken from the Top of Stack. The source operand must be expressed in radians and must be in the range $|x| < 2^{63}$. FPTAN rounds the result (according to the mode in effect), places the value onto the Top of Stack, then pushes 1.0 onto the Top of Stack.

Condition Codes: C0 and C3 are undefined. C2 specifies reduction, where C2 = 1 means the reduction is incomplete. In case of a register error, C1 indicates the type of error, where C1 = 1 means the destination register is full and C1 = 0 means the source register is empty. C1 also indicates the type of rounding (away from 0) after a Precision exception.

Zero/Infinity:

Operand 1 (x)	Result	Operand 1 (x)	Result
+0	$y=+0$	$+\infty$	Invalid operation
-0	$y=-0$	$-\infty$	Invalid operation

T-49-12-05

Exception Flags:

Exception	Mode	Result	S	P	U	O	Z	D	I
Register Error	Masked	QNaN	1						1
	Unmasked	Unchanged	1						1
Precision	Masked	Rounded		1					
	Unmasked	Rounded		1					
Underflow	Masked	Denormal/0		1	1				
	Unmasked	Round, scale			1				
Denormal	Masked	Denormal used						1	
	Unmasked	Trap, abort						1	
Invalid Operation	Masked	QNaN							1
	Unmasked	Unchanged							1

T-49-12-05

FRNDINT — Round to integer

Syntax: **FRNDINT**

Format: **FRNDINT**

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FRNDINT	ST(0)	D9h	1	1	1	1	1	1	0	0	5-110	

The FRNDINT instruction rounds the value in the Top of Stack to an integer. The rounding operation performed depends on the value of the RC field in the Control Word.

Condition Codes: C0, C2, and C3 are undefined. C1 indicates whether the source operand was rounded up (C1 = 1) or rounded down (C1 = 0). C1 is also set to 0 if the source register is empty.

Zero/Infinity:

Operand 1	Result	Operand 1	Result
+0	+0	+∞	+∞
-0	-0	-∞	-∞

T-49-12-05

Exception Flags:

Exception	Mode	Result	S	P	U	O	Z	D	I
Register Error	Masked	QNaN	1						1
	Unmasked	Unchanged	1						1
Precision	Masked	Rounded		1					
	Unmasked	Rounded		1					
Denormal	Masked	Denormal used	0					1	
	Unmasked	Trap, abort	0					1	
Invalid Operation	Masked	QNaN							1
	Unmasked	Unchanged							1

T-49-12-05

FRSTOR — Restore FPU state

Syntax: **FRSTOR** *dest*

Format: **FRSTOR** *memory*

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FRSTOR	mem94/108byte	DDh	MD1	MD0	1	0	0	RM2	RM1	RM0	SIB,displ	106

The FRSTOR instruction loads the SuperMathSX FPU environment and registers from the memory location specified by the source operand. The data structure of the environment depends on the operand size and the current operating mode. This instruction is normally executed in conjunction with FSAVE, which saves the environment and registers to a specific memory location.

The SuperMathSX FPU environment is determined by the Mode Control Word, the Status Word, the Tag Word, the Instruction Pointer, and the Data Pointer.

FRSTOR loads the registers after loading the environment. The registers are in the 80 bytes following the environment data.

Condition Codes: C0, C1, C2, and C3 are loaded from memory.

Exception Flags: Exceptions may be loaded with the environment. If the Status Word loaded with the environment contains an exception that is enabled, an error trap will occur after the next wait or after the execution of an Exception Status instruction.

T-49-12-05

FSAVE — Save FPU state

Syntax: **FSAVE** *dest*

Format: **FSAVE** *memory*

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FSAVE	mem94/108byte	DDh	MD1	MD0	1	1	0	RM2	RM1	RM0	SIB,displ	106

The FSAVE instruction saves the SuperMathSX FPU environment and registers to the memory location specified by the source operand.

The SuperMathSX FPU environment is made up of the Mode Control Word, the Status Word, the Data Register Tag Word, the Instruction Pointer, and the Data Pointer.

FSAVE saves the registers in memory after the storing the environment. The registers are stored in the 80 bytes following the environment data.

The current 80386SX operating mode and operand size determine the format of the environment, as shown in Figures 20, 21, 22, and 23.

T-49-12-05

Figure 20. 16-Bit Protected Mode

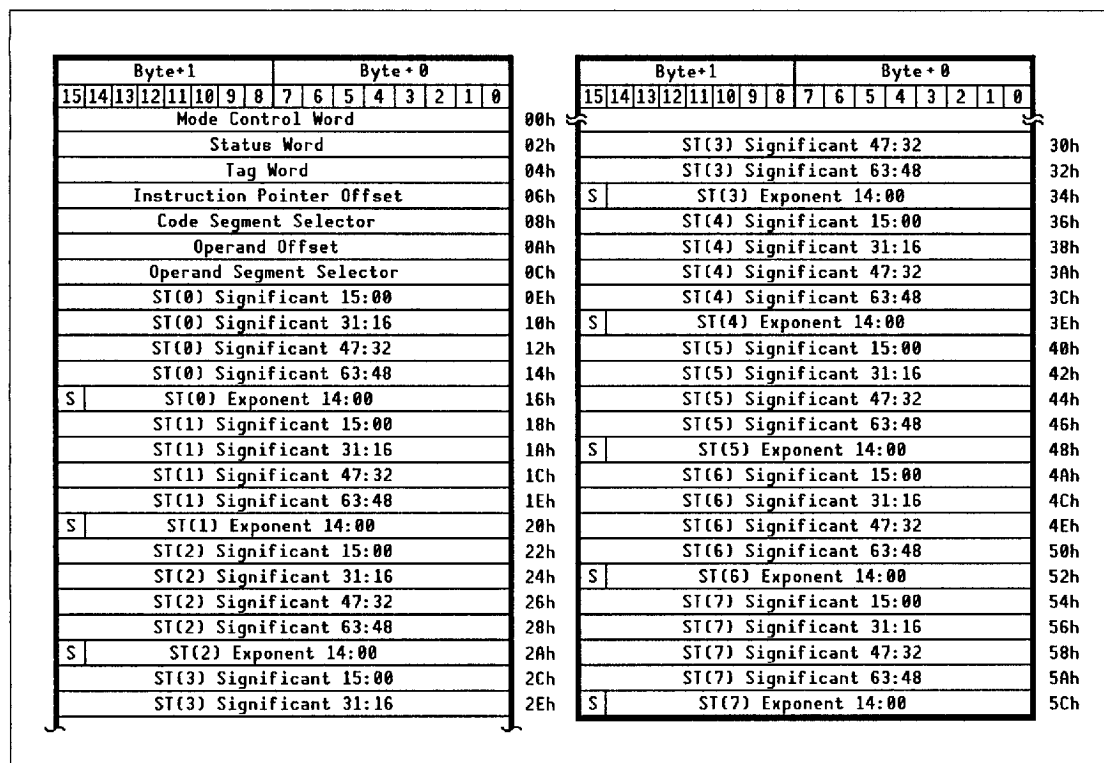


Figure 21. *16-Bit Real Mode*

T-49-12-05

Byte+1								Byte+0								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Mode Control Word																00h
Status Word																02h
Tag Word																04h
Instruction Pointer 15:00																06h
IP 19:16		0		Opcode 10:00												08h
Operand Pointer 15:00																0Ah
OP 19:16		0		0		0		0		0		0		0		0
ST(0) Significant 15:00																0Ch
ST(0) Significant 31:16																10h
ST(0) Significant 47:32																12h
ST(0) Significant 63:48																14h
S		ST(0) Exponent 14:00														16h
ST(1) Significant 15:00																18h
ST(1) Significant 31:16																1Ah
ST(1) Significant 47:32																1Ch
ST(1) Significant 63:48																1Eh
S		ST(1) Exponent 14:00														20h
ST(2) Significant 15:00																22h
ST(2) Significant 31:16																24h
ST(2) Significant 47:32																26h
ST(2) Significant 63:48																28h
S		ST(2) Exponent 14:00														2Ah
ST(3) Significant 15:00																2Ch
ST(3) Significant 31:16																2Eh

Byte+1								Byte+0								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
ST(3) Significant 47:32																30h
ST(3) Significant 63:48																32h
S		ST(3) Exponent 14:00														34h
ST(4) Significant 15:00																36h
ST(4) Significant 31:16																38h
ST(4) Significant 47:32																3Ah
ST(4) Significant 63:48																3Ch
S		ST(4) Exponent 14:00														3Eh
ST(5) Significant 15:00																40h
ST(5) Significant 31:16																42h
ST(5) Significant 47:32																44h
ST(5) Significant 63:48																46h
S		ST(5) Exponent 14:00														48h
ST(6) Significant 15:00																4Ah
ST(6) Significant 31:16																4Ch
ST(6) Significant 47:32																4Eh
ST(6) Significant 63:48																50h
S		ST(6) Exponent 14:00														52h
ST(7) Significant 15:00																54h
ST(7) Significant 31:16																56h
ST(7) Significant 47:32																58h
ST(7) Significant 63:48																5Ah
S		ST(7) Exponent 14:00														5Ch

T-49-12-05

Figure 22. 32-Bit Protected Mode

Byte+3								Byte+2								Byte+1								Byte+0								
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved																Mode Control Word																00h
Reserved																Status Word																04h
Reserved																Tag Word																08h
Instruction Pointer Offset																																0Ch
0	0	0	0	0	0	Opcode 10:00										Code Segment Selector																10h
Operand Offset																																14h
Reserved																Operand Segment Selector																18h
ST(0) Significand 31:00																																1Ch
ST(0) Significand 63:32																																20h
ST(1) Significand 15:00																S	ST(0) Exponent 14:00															24h
ST(1) Significand 47:16																																28h
S	ST(1) Exponent 14:00															ST(1) Significand 63:48																2Ch
ST(2) Significand 31:00																																30h
ST(2) Significand 63:32																																34h
ST(3) Significand 15:00																S	ST(2) Exponent 14:00															38h
ST(3) Significand 47:16																																3Ch
S	ST(3) Exponent 14:00															ST(3) Significand 63:48																40h
ST(4) Significand 31:00																																44h
ST(4) Significand 63:32																																48h
ST(5) Significand 15:00																S	ST(4) Exponent 14:00															4Ch
ST(5) Significand 47:16																																50h
S	ST(5) Exponent 14:00															ST(5) Significand 63:48																54h
ST(6) Significand 31:00																																58h
ST(6) Significand 63:32																																5Ch
ST(7) Significand 15:00																S	ST(6) Exponent 14:00															60h
ST(7) Significand 47:16																																64h
S	ST(7) Exponent 14:00															ST(7) Significand 63:48																68h

Figure 23. 32-Bit Real Mode

T-49-12-05

Byte+3								Byte+2								Byte+1								Byte+0											
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
Reserved																Mode Control Word																00h			
Reserved																Status Word																04h			
Reserved																Tag Word																08h			
Reserved																Instruction Pointer 15:00																0Ch			
0	0	0	0	Instruction Pointer 31:16												0	Opcode 10:00										10h								
Reserved																Operand Pointer 15:00																14h			
0	0	0	0	Operand Pointer 31:16												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18h
ST(0) Significand 31:00																																1Ch			
ST(0) Significand 63:32																																20h			
ST(1) Significand 15:00																S	ST(0) Exponent 14:00																24h		
ST(1) Significand 47:16																																28h			
S	ST(1) Exponent 14:00															ST(1) Significand 63:48																2Ch			
ST(2) Significand 31:00																																30h			
ST(2) Significand 63:32																																34h			
ST(3) Significand 15:00																S	ST(2) Exponent 14:00																38h		
ST(3) Significand 47:16																																3Ch			
S	ST(3) Exponent 14:00															ST(3) Significand 63:48																40h			
ST(4) Significand 31:00																																44h			
ST(4) Significand 63:32																																48h			
ST(5) Significand 15:00																S	ST(4) Exponent 14:00																4Ch		
ST(5) Significand 47:16																																50h			
S	ST(5) Exponent 14:00															ST(5) Significand 63:48																54h			
ST(6) Significand 31:00																																58h			
ST(6) Significand 63:32																																5Ch			
ST(7) Significand 15:00																S	ST(6) Exponent 14:00																60h		
ST(7) Significand 47:16																																64h			
S	ST(7) Exponent 14:00															ST(7) Significand 63:48																68h			

Condition Codes: C0, C1, C2, and C3 are all set to 0.

Exception Flags: No exceptions.

T-49-12-05

FSCALE— Multiply by 2ⁿ

Syntax: **FSCALE**

Format: **FSCALE**

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FSCALE	ST(0)	D9h	1	1	1	1	1	1	0	1	11	

The FSCALE instruction multiplies the contents of the Top of Stack by 2ⁿ. (The value for n is the next to Top of Stack value truncated toward 0 to convert it into an integer.) The result is rounded according to the mode in effect and placed in the Top of Stack.

Condition Codes: C0, C2, and C3 are undefined. C1 is set to 0 except after a Precision exception, when it signifies whether rounding was towards or away from 0.

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002684 837 ■ CHP

Zero/Infinity:

T-49-12-05

x	n	Result
+0	$-\infty$	+0
-0	$-\infty$	-0
0	$+\infty$	Invalid operation
$+\infty$	$-\infty$	Invalid operation
$-\infty$	$-\infty$	Invalid operation
$+\infty$	$-\infty < n \leq +\infty$	$+\infty$
$-\infty$	$-\infty < n \leq -\infty$	$-\infty$
$x \neq 0$	$+\infty$	$\text{sign}(x) * \infty$
$x \neq 0$	$-\infty$	$\text{sign}(x) * 0$

Exception Flags:

Exception	Mode	Result	S	P	U	O	Z	D	I
Register Error	Masked	QNaN	1						1
	Unmasked	Unchanged	1						1
Precision	Masked	Rounded		1					
	Unmasked	Rounded		1					
Underflow	Masked	Denormal/0		1	1				
	Unmasked	Round, scale			1				
Overflow	Masked	Infinity				1			
	Unmasked	Round, scale				1			
Denormal	Masked	Denorm						1	
	Unmasked	Trap, abort						1	
Invalid Operation	Masked	QNaN							1
	Unmasked	Unchanged							1

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002685 773 ■ CHP

FSIN — Function evaluation of sine(x)

T-49-12-05

Syntax: FSIN

Format: FSIN

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FSIN	ST(0)	D9h	1	1	1	1	1	1	1	0	5-126	

The FSIN instruction calculates the sine of x , which is the source operand taken from the Top of Stack. The source operand must be expressed in radians and must be in the range $|x| < 2^{63}$. FSIN rounds the result (according to the mode in effect) and places the value onto the Top of Stack.

Condition Codes: C0 and C3 are undefined. C2 specifies reduction where C2 = 1 means the reduction is incomplete. In case of a Precision exception, C1 indicates the type of rounding (away from 0).

Zero/Infinity:

Operand	Result	Operand	Result
+0	+0	$+\infty$	Invalid operation
-0	-0	$-\infty$	Invalid operation

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002686 60T ■ CHP

Exception Flags:

T-49-12-05

Exception	Mode	Result	S	P	U	O	Z	D	I
Register Error	Masked	QNaN	1						1
	Unmasked	Unchanged	1						1
Precision	Masked	Rounded		1					
	Unmasked	Rounded		1					
Underflow	Masked	Denormal/0		1	1				
	Unmasked	Round, scale			1				
Denormal	Masked	Denorm used						1	
	Unmasked	Trap, abort						1	
Invalid Operation	Masked	QNaN							1
	Unmasked	Unchanged							1

FSINCOS— Function evaluation of sine(x) and cosine(y)

T-49-12-05

Syntax: FSINCOS
Format: FSINCOS
Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FSINCOS	ST(0)	D9h	1	1	1	1	1	0	1	1	5-237	

The FSINCOS instruction calculates the sine of x and the cosine of x. The source operand, x, is taken from the Top of Stack. The source operand must be expressed in radians and must be in the range $|x| < 2^{63}$. FSINCOS rounds the results (according to the mode in effect); places y, the result of sin(x), onto the top of stack; then pushes z, the result of cos(x), onto the stack.

Condition Codes: C0 and C3 are undefined. C2 specifies reduction, where C2 = 1 means the reduction is incomplete. In case of a register error, C1 indicates the type of error, where C1 = 1 means the destination register is full and C1 = 0 means the source register is empty. C1 also indicates the type of rounding (away from 0) after a Precision exception.

Zero/Infinity:

Operand (x)	Result	Operand (x)	Result
+0	y=+0, z=+1	+∞	Invalid operation
-0	y=-0, z=+1	-∞	Invalid operation

T-49-12-05

Exception Flags:

Exception	Mode	Result	S	P	U	O	Z	D	I
Register Error	Masked	QNaN	1						1
	Unmasked	Unchanged	1						1
Precision	Masked	Rounded		1					
	Unmasked	Rounded		1					
Underflow	Masked	Denormal/0		1	1				
	Unmasked	Round, scale			1				
Denormal	Masked	Denorm used						1	
	Unmasked	Trap, abort						1	
Invalid Operation	Masked	QNaN							1
	Unmasked	Unchanged							1

FSQRT — Calculate the square root of x

T-49-12-05

Syntax: **FSQRT**

Format: **FSQRT**

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FSQRT	ST(0)	D9h	1	1	1	1	1	0	1	0	19-39	

The FSQRT instruction calculates the square root of x. The source operand, x, is taken from the Top of Stack. FSQRT rounds the results to the appropriate precision (according to the mode in effect) and puts the result onto the Top of Stack.

Condition Codes: C0, C2, and C3 are undefined. C1 indicates the type of rounding (away from 0) after a Precision exception but is otherwise 0.

Zero/Infinity:

Operand (x)	Result	Operand (x)	Result
+0	+0	$-\infty \leq x < -0$	Invalid operation
-0	-0	$+\infty$	$+\infty$

T-49-12-05

Exception Flags:

Exception	Mode	Result	S	P	U	O	Z	D	I
Register Error	Masked	QNaN	1						1
	Unmasked	Unchanged	1						1
Precision	Masked	Rounded		1					
	Unmasked	Rounded		1					
Denormal	Masked	Denorm used						1	
	Unmasked	Trap, abort						1	
Invalid Operation	Masked	QNaN							1
	Unmasked	Unchanged							1

FST — Store register

T-49-12-05

Syntax: **FST(P)** *dest*

Format: **FIST(P)** *memory*
FBSTP *memory*
FST(P) *memory*
FST(P) *register*

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding										Clock Cycles
		1st Byte	2nd Byte								Bytes 3-7	
			B7	B6	B5	B4	B3	B2	B1	B0		
FIST	mem16i,ST(0)	DFh	MD1	MD0	0	1	0	RM2	RM1	RM0	SIB,displ	8
FISTP	mem16i,ST(0)	DFh	MD1	MD0	0	1	1	RM2	RM1	RM0	SIB,displ	8
FIST	mem32i,ST(0)	DBh	MD1	MD0	0	1	0	RM2	RM1	RM0	SIB,displ	8
FISTP	mem32i,ST(0)	DBh	MD1	MD0	0	1	1	RM2	RM1	RM0	SIB,displ	8
FISTP	mem64i,ST(0)	DFh	MD1	MD0	1	1	1	RM2	RM1	RM0	SIB,displ	8
FBSTP	mem80b,ST(0)	DFh	MD1	MD0	1	1	0	RM2	RM1	RM0	SIB,displ	67
FST	mem32r,ST(0)	D9h	MD1	MD0	0	1	0	RM2	RM1	RM0	SIB,displ	8
FSTP	mem32r,ST(0)	D9h	MD1	MD0	0	1	1	RM2	RM1	RM0	SIB,displ	8
FST	mem64r,ST(0)	DDh	MD1	MD0	0	1	0	RM2	RM1	RM0	SIB,displ	9
FSTP	mem64r,ST(0)	DDh	MD1	MD0	0	1	1	RM2	RM1	RM0	SIB,displ	9
FSTP	mem80r,ST(0)	DBh	MD1	MD0	1	1	1	RM2	RM1	RM0	SIB,displ	4
FST	ST(n),ST(0)	DDh	1	1	0	1	0	REG2	REG1	REG0		3
FSTP	ST(n),ST(0)	DDh	1	1	0	1	1	REG2	REG1	REG0		3

The FST instruction takes the value from the Top of Stack, converts the data format and rounds it, if necessary, and puts it into the destination register or memory location. The forms FISTP, FSTP, and FBSTP copy the value in the Top of Stack, then pop the stack.

The source operand is rounded to fit the size of the destination according to the mode in effect. If the source operand is a 0, ∞ , or NaN, it is chopped on the right (not rounded) to fit the destination.

T-49-12-05

Condition Codes: C0, C2, and C3 are undefined. C1 indicates the type of rounding (away from 0) after a Precision exception but is otherwise 0.

Zero/Infinity:

Operand (x)	Result	Operand (x)	Result
Empty	Invalid operation	$\infty \rightarrow \text{Integer}$	Invalid operation
$\text{NaN} \rightarrow \text{Integer}$	Invalid operation	$ x > \text{Integer range}$	Invalid operation

Exception Flags:

Exception	Mode	Result	S	P	U	O	Z	D	I
Register Error	Masked	QNaN	1						1
	Unmasked	Unchanged	1						1
Precision	Masked	Rounded		1					
	Unmasked	Rounded		1					
Underflow	Masked	Rounded		1	1				
	Unmasked	Trap, abort			1				
Overflow	Masked	Rounded				1			
	Unmasked	Trap, abort				1			
Invalid Operation	Masked	QNaN							1
	Unmasked	Trap, abort							1

FSTCW — Store control word

T-49-12-05

Syntax: FSTCW *dest*

Format: FSTCW *memory*

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles
		1st Byte	2nd Byte							Bytes 3-7	
			B7	B6	B5	B4	B3	B2	B1		
FSTCW	mem2byte	D9h	MD1	MD0	1	1	1	RM2	RM1	RM0	5

The FSTCW instruction stores the contents of the Mode Control Word in the specified destination in memory.

Condition Codes: C0, C1, C2, and C3 are undefined.

Exception Flags: No exceptions.

T-49-12-05

FSTENV — Store environment

Syntax: **FSTENV** *dest*

Format: **FSTENV** *memory*

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FSTENV	mem14/28byte	D9h	MD1	MD0	1	1	0	RM2	RM1	RM0	SIB,displ	56

The FSTENV instruction saves the SuperMathSX FPU environment to the specified memory location. The 80386SX mode and operand size determine the format of the environment as shown below. The environment is made up of the Mode Control Word, the Status Word, the Data Register Tag Word, the Instruction Pointer, and the Data Pointer.

Condition Codes: C0, C1, C2, and C3 are undefined.

Exception Flags: No exceptions.

Figure 24. *SuperMathSX Coprocessor Environment*

T-49-12-05

16-Bit Protected Mode

Byte+1								Byte+0								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Mode Control Word																00h
Status Word																02h
Tag Word																04h
Instruction Pointer Offset																06h
Code Segment Selector																08h
Operand Offset																0Ah
Operand Segment Selector																0Ch

16-Bit Real Mode

[illegible]

32-Bit Protected Mode

Byte+3					Byte+2					Byte+1					Byte+0																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved																Mode Control Word												00h				
Reserved																Status Word												04h				
Reserved																Tag Word												08h				
Instruction Pointer Offset																																0Ch
0	0	0	0	0	0	Opcode 10:0																Code Segment Selector										10h
Operand Offset																																14h
Reserved																Operand Segment Selector																18h

32-Bit Real Mode

Byte+3								Byte+2								Byte+1								Byte+0																					
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
Reserved																Mode Control Word																00h													
Reserved																Status Word																04h													
Reserved																Tag Word																08h													
Reserved																Instruction Pointer 15:00																0Ch													
0				0				0				0				Instruction Pointer 31:16								0	Opcode 10:00								10h												
Reserved																Operand Pointer 15:00																14h													
0				0				0				0				Operand Pointer 31:16								0	0				0				0				0				0				18h

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002696 559 ■ CHP

FSTSW — Store status word

T-49-12-05

Syntax: **FSTSW** *dest*

Format: **FSTSW** *memory*

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FSTSW	mem2byte	DDh	MD1	MD0	1	1	1	RM2	RM1	RM0		5
FSTSW	80386SX Reg	DFh	1	1	1	1	1	0	0	0		5

The FSTSW instruction stores the contents of the Status Word in the specified destination in memory or in the AX register of the 80386SX compatible processor.

Condition Codes: C0, C1, C2, and C3 are undefined.

Exception Flags: No exceptions.

T-49-12-05

FSUB — Floating-point subtraction

Syntax: **FSUB(R)(P) [[*dest*,] *source*]**

Format: **FISUB(R) *stack, memory***
FSUB(R) *stack, memory*
FSUB(R) *stack, register*
FSUB(R)(P) *register, stack*

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FISUB	ST(0),mem16i	DEh	MD1	MD0	1	0	0	RM2	RM1	RM0	SIB,displ	13-15
FISUBR	ST(0),mem16i	DEh	MD1	MD0	1	0	1	RM2	RM1	RM0	SIB,displ	13-15
FISUB	ST(0),mem32i	DAh	MD1	MD0	1	0	0	RM2	RM1	RM0	SIB,displ	17
FISUBR	ST(0),mem32i	DAh	MD1	MD0	1	0	1	RM2	RM1	RM0	SIB,displ	17
FSUB	ST(0),mem32r	D8h	MD1	MD0	1	0	0	RM2	RM1	RM0	SIB,displ	10-15
FSUBR	ST(0),mem32r	D8h	MD1	MD0	1	0	1	RM2	RM1	RM0	SIB,displ	10-15
FSUB	ST(0),mem64r	DCh	MD1	MD0	1	0	0	RM2	RM1	RM0	SIB,displ	14-17
FSUBR	ST(0),mem64r	DCh	MD1	MD0	1	0	1	RM2	RM1	RM0	SIB,displ	14-17
FSUB	ST(0),ST(n)	D8h	1	1	1	0	0	REG2	REG1	REG0		7-9
FSUBR	ST(0),ST(n)	D8h	1	1	1	0	1	REG2	REG1	REG0		7-9
FSUB	ST(n),ST(0)	DCh	1	1	1	0	1	REG2	REG1	REG0		7-9
FSUBR	ST(n),ST(0)	DCh	1	1	1	0	0	REG2	REG1	REG0		6-7
FSUBP	ST(n),ST(0)	DEh	1	1	1	0	1	REG2	REG1	REG0		6-7
FSUBRP	ST(n),ST(0)	DEh	1	1	1	0	0	REG2	REG1	REG0		6

The source operand is subtracted from the destination. The result is rounded (according to the mode in effect) and returned to the destination.

When using either of the pop forms (FSUBP or FSUBRP), the instruction pops the ST(0). The reverse forms (FISUBR, FSUBR, or FSUBRP) subtract the destination from the source.

Condition Codes: C0, C2, and C3 are undefined. C1 is set to 0 except after a Precision exception, where it defines whether or not rounding is away from 0.

T-49-12-05

Zero/Infinity:

Operand 1	Operand 2	Result	Operand 1	Operand 2	Result
+0	+0	R(0)	$-\infty$	$-\infty$	Invalid
-0	-0	R(0)	$+\infty$	$+\infty$	$+\infty$
+0	-0	+0	$-\infty$	$-\infty$	$-\infty$
-0	+0	-0	$+\infty$	X	$+\infty$
+X	+X	R(0)	$-\infty$	X	$-\infty$
-X	-X	R(0)	X	$-\infty$	$+\infty$
$+\infty$	$+\infty$	Invalid	X	$+\infty$	$-\infty$

Exception Flags:

Exception	Mode	Result	S	P	U	O	Z	D	I
Register	Masked	QNaN	1						1
Error	Unmasked	Unchanged	1						1
Precision	Masked	Rounded		1					
	Unmasked	Rounded		1					
Underflow	Masked	Denormal/0		1	1				
	Unmasked	Round, scale			1				
Overflow	Masked	Infinity				1			
	Unmasked	Round, scale				1			
Denormal	Masked	Denorm used						1	
	Unmasked	Trap, abort						1	
Invalid	Masked	QNaN							1
Operation	Unmasked	Unchanged							1

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002699 268 ■ CHP

FTST— Test top of stack

T-49-12-05

Syntax: **FTST**Format: **FTST**

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FTST	ST(0)	D9h	1	1	1	0	0	1	0	0	3	

The FTST instruction compares the value in the Top of Stack to 0. The condition codes reflect the result of the comparison.

Condition Codes: The result of the comparison is determined by the condition code bits C3, C2, C1, and C0, as follows:

ST(0) Status	C3	C2	C1	C0
ST(0) >0	0	0	0	0
ST(0) <0	0	0	0	1
ST(0) = +0	1	0	0	0
ST(0) = -0	1	0	0	0
Unordered	1	1	0	1

If the source operand is either a NaN or an unsupported number, or if a stack fault occurs, an Invalid exception occurs and the condition code bits are undefined.

If the source register is empty, then C0, C2, and C3 are undefined; C1 = 0.

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002700 80T ■ CHP

Zero/Infinity:

T-49-12-05

ST(0)	Result	ST(0)	Result
+0	=0	+∞	>0
-0	-0	-∞	<0

Exception Flags:

Exception	Mode	Result	S	P	U	O	Z	D	I
Register Error	Masked	Unchanged	1						1
	Unmasked	Unchanged	1						1
Denormal	Masked	Denorm used						1	
	Unmasked	Denorm used						1	
Invalid* Operation	Masked	Unchanged							1
	Unmasked	Unchanged							1

* QNaNs cause an Invalid Operation exception to occur.

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002701 746 ■ CHP

FUCOM— Unordered compare

T-49-12-05

Syntax: **FUCOM(P) [[*dest*,] *source*]**Format: **FUCOM(P)(P) *stack,register***

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FUCOM	ST(n),ST(0)	DDh	1	1	1	0	0	REG2	REG1	REG0		4
FUCOMP	ST(n),ST(0)	DDh	1	1	1	0	1	REG2	REG1	REG0		5
FUCOMPP	ST(1),ST(0)	DAh	1	1	0	1	1	0	0	1		7

The FUCOM instruction compares the source to the Top of Stack. The condition codes reflect the result of the comparison. FUCOMP pops the Top of Stack. FUCOMPP compares the Top of Stack to the next to Top of Stack and then pops the Top of Stack twice.

Condition Codes: The result of the comparison is determined by the condition code bits C3, C2, C1, C0, as follows:

<i>dest/source</i> Status	C3	C2	C1	C0
<i>dest</i> > <i>source</i>	0	0	0	0
<i>dest</i> < <i>source</i>	0	0	0	1
<i>dest</i> = <i>source</i>	1	0	0	0
Unordered	1	1	0	1

If the source operand is either a SNAN or an unsupported number, or if a stack fault occurs, an Invalid exception occurs and the condition code bits are undefined.

If the source register is empty, C3 = C2 = C0 = 1 and C1 = 0.

Zero/Infinity:

T-49-12-05

dest	source	Result	dest	source	Result
+0	+0	=	+∞	+∞	=
-0	-0	=	-∞	-∞	=
+0	-0	=	+∞	-∞	dest>src
-0	+0	=	-∞	+∞	dest<src
+0	+x	dest<src	+∞	x	dest>src
-0	+x	dest<src	-∞	x	dest<src
+0	-x	dest>src	x	-∞	dest>src
-0	-x	dest>src	x	+∞	dest<src

Exception Flags:

Exception	Mode	Result	S	P	U	O	Z	D	I
Register Error	Masked	Unordered	1						1
	Unmasked	Unchanged	1						1
Denormal	Masked	Denorm used						1	
	Unmasked	Denorm used						1	
Invalid Operation	Masked	Unordered						1	
	Unmasked	Unchanged						1	

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002703 519 ■ CHP

FXAM— Examine operand

T-49-12-05

Syntax: **FXAM**Format: **FXAM**

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FXAM	ST(0)	D9h	1	1	1	0	0	1	0	1	3	

The FXAM instruction examines the source operand (Top of Stack) and reports its type by setting the condition codes.

Condition Codes: The condition codes are set to report the result of the examination.

ST(0) Contents	C3	C2	C0
Unnormal	0	0	0
NaN	0	0	1
Normal	0	1	0
Infinity	0	1	1
Zero	1	0	0
Empty	1	0	1
Denormal	1	1	0

C1 reflects the sign of the source operand, where 0 = positive and 1 = negative.

Exception Flags: No exceptions.

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002704 455 ■ CHP

FXCH— Exchange register contents

T-49-12-05

Syntax: **FXCH**

Format: **FXCH**

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FXCH	ST(n),ST(0)	D9h	1	1	0	0	1	REG2	REG1	REG0	4	

The FXCH instruction swaps the contents of the Top of Stack and the source register.

Condition Codes: C1 is always 0. C0, C2, and C3 are undefined after normal execution.

Exception Flags:

Exception	Mode	Result	S	P	U	O	Z	D	I
Register	Masked	Unordered	1						1
Error	Unmasked	Unchanged	1						1

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002705 391 ■ CHP

FXTRACT — Extract exponent

T-49-12-05

Syntax: **FXTRACT**Format: **FXTRACT**

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FXTRACT	ST(0)	D9h	1	1	1	1	0	1	0	0	6	

The FXTRACT instruction operates on the value contained in ST(0). The exponent portion of ST(0) is converted into 80-bit extended precision format and is pushed onto the stack (y). The exponent of x is set to 0 (i.e., 3FFh biased) but its sign is preserved.

Condition Codes: C0, C2, and C3 are undefined. C1 is set to 0 after normal execution. In case of a register error, C1 indicates the type of error, where C1 = 1 means the destination register is full and C1 = 0 means the source register is empty.

Zero/Infinity:

Operand 1	Result
+0	y=+0, x=-∞, Divide by Zero exception
-0	y=-0, x=-∞, Divide by Zero exception
+∞	y=-∞, x=+∞
-∞	y=+∞, x=+∞

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002706 228 ■ CHP

Exception Flags:

T-49-12-05

Exception	Mode	Result	S	P	U	O	Z	D	I
Register Error	Masked	QNaN	1						1
	Unmasked	Unchanged	1						1
Divide by Zero	Masked	ST(0)=0, ST(1)=-∞;					1		
	Unmasked	Trap, abort					1		
Denormal	Masked	Denormal used						1	
	Unmasked	Trap, abort						1	
Invalid Operation	Masked	QNaN							1
	Unmasked	Trap, unchanged							1

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002707 164 ■ CHP

FYL2X— Compute $y * \log_2(x)$

T-49-12-05

Syntax: **FYL2X**Format: **FYL2X**

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding										Clock Cycles
		1st Byte	2nd Byte								Bytes 3-7	
			B7	B6	B5	B4	B3	B2	B1	B0		
FYL2X	ST(0)	D9h	1	1	1	1	0	0	0	1	12-236	

The FYL2X instruction computes the base 2 logarithm of x, which is the value in the Top of Stack. It then multiplies the result by the next to Top of Stack value, y, rounds the result according to the mode in effect, places the result in the next to Top of Stack, then pops the stack.

If x is negative, the invalid operation exception occurs.

Condition Codes: C0, C2, and C3 are undefined. C1 indicates the type of rounding (away from 0) after a Precision exception but is otherwise 0.

CHIPS & TECHNOLOGIES INC SLE D ■ 2098116 0002708 070 ■ CHP

Zero/Infinity:

T-49-12-05

x	y	Result
$x < 0$		Invalid operation
$x = 0$	$y \neq 0$	Divide by Zero exception
$x = 0$	$y = 0$	Invalid operation
$x = 1$	$y = \infty$	Invalid operation
$x > 1$	$y = \infty$	y
$0 < x < 1$	$y = \infty$	-y
$x = \infty$	$y > +0$	$+\infty$
$x = \infty$	$y < -0$	$-\infty$
$x = \infty$	$y = 0$	Invalid operation

Exception Flags:

Exception	Mode	Result	S	P	U	O	Z	D	I
Register Error	Masked	QNaN	1						1
	Unmasked	Unchanged	1						1
Precision	Masked	Rounded		1					
	Unmasked	Rounded		1					
Underflow	Masked	Denormal/0		1	1				
	Unmasked	Round, scale			1				
Overflow	Masked	Infinity				1			
	Unmasked	Round, scale				1			
Divide by Zero	Masked	ST(0) = $-\infty$					1		
	Unmasked	Trap, abort					1		
Denormal	Masked	Denorm used						1	
	Unmasked	Trap, abort						1	
Invalid Operation	Masked	QNaN							1
	Unmasked	Unchanged							1

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002709 T37 ■ CHP

FYL2XP1 — Compute $y * \log_2(x+1)$

T-49-12-05

Syntax: **FYL2XP1**Format: **FYL2XP1**

Description:

Instruction	Operand(s) (dest, source)	Opcode/Instruction Encoding									Clock Cycles	
		1st Byte	2nd Byte									Bytes 3-7
			B7	B6	B5	B4	B3	B2	B1	B0		
FYL2XP1	ST(0)	D9h	1	1	1	1	1	0	0	1	5-227	

The FYL2XP1 instruction computes the base 2 logarithm of x (the Top of Stack) plus 1.0. It then multiplies the result by the next to Top of Stack value, y , rounds the result according to the mode in effect, places the result in the next to Top of Stack, then pops the stack.

FYL2XP1 is used to compute the logarithm of numbers whose absolute value is close to 1. FYL2XP1 is more accurate than FYL2X in this case. The operand, x , (the Top of Stack) must be in the range $-(1-\text{SQRT}(2))/2 < x < 1-\text{SQRT}(2)/2$.

Condition Codes: C0, C2, and C3 are undefined. C1 indicates the type of rounding (away from 0) after a Precision exception but is otherwise 0.

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002710 759 ■ CHP

Zero/Infinity:

T-49-12-05

x	y	Result
$x = -0$	$y \geq +0$	-0
$x = +0$	$y \geq +0$	$+0$
$x = -0$	$y \leq -0$	$+0$
$x = +0$	$y < -0$	-0
$x = 0$	$y = \infty$	Invalid operation
$x > 0$	$y = \infty$	y
$-1 < x < 0$	$y = \infty$	$-y$
$x = \infty$	$y > +0$	$+\infty$
$x = \infty$	$y < -0$	$-\infty$
$x = \infty$	$y = 0$	Invalid operation

Exception Flags:

Exception	Mode	Result	S	P	U	O	Z	D	I
Register Error	Masked	QNaN	1						1
	Unmasked	Unchanged	1						1
Precision	Masked	Rounded		1					
	Unmasked	Rounded		1					
Underflow	Masked	Denormal/0		1	1				
	Unmasked	Round, scale			1				
Denormal	Masked	Denorm used						1	
	Unmasked	Trap, abort						1	
Invalid Operation	Masked	QNaN							1
	Unmasked	Unchanged							1

SuperMathSX Operands

T-49-12-05

Table 18 summarizes the operands and shows the encoding for all SuperMath FPU instructions.

Table 18. SuperMathSX Operands

Opcode/Instruction Encoding										Instruction	Operand(s) (dest,source)
1st Byte	2nd Byte								Bytes 3-7		
	B7	B6	B5	B4	B3	B2	B1	B0			
D8h	1	1	0	0	0	REG2	REG1	REG0		FADD	ST(0), ST(n)
D8h	1	1	0	0	1	REG2	REG1	REG0		FMUL	ST(0), ST(n)
D8h	1	1	0	1	0	REG2	REG1	REG0		FCOM	ST(n)
D8h	1	1	0	1	1	REG2	REG1	REG0		FCOMP	ST(n)
D8h	1	1	1	0	0	REG2	REG1	REG0		FSUB	ST(0), ST(n)
D8h	1	1	1	0	1	REG2	REG1	REG0		FSUBR	ST(0), ST(n)
D8h	1	1	1	1	0	REG2	REG1	REG0		FDIV	ST(0), ST(n)
D8h	1	1	1	1	1	REG2	REG1	REG0		FDIVR	ST(0), ST(n)
D8h	MD1	MD0	0	0	0	RM2	RM1	RM0	SIB,DISPL	FADD	ST(0), mem32r
D8h	MD1	MD0	0	0	1	RM2	RM1	RM0	SIB,DISPL	FMUL	ST(0), mem32r
D8h	MD1	MD0	0	1	0	RM2	RM1	RM0	SIB,DISPL	FCOM	ST(0), mem32r
D8h	MD1	MD0	0	1	1	RM2	RM1	RM0	SIB,DISPL	FCOMP	ST(0), mem32r
D8h	MD1	MD0	1	0	0	RM2	RM1	RM0	SIB,DISPL	FSUB	ST(0), mem32r
D8h	MD1	MD0	1	0	1	RM2	RM1	RM0	SIB,DISPL	FSUBR	ST(0), mem32r
D8h	MD1	MD0	1	1	0	RM2	RM1	RM0	SIB,DISPL	FDIV	ST(0), mem32r
D8h	MD1	MD0	1	1	1	RM2	RM1	RM0	SIB,DISPL	FDIVR	ST(0), mem32r
D9h	1	1	0	0	0	REG2	REG1	REG0		FLD	ST(n)
D9h	1	1	0	0	1	REG2	REG1	REG0		FXCH	ST(n)
D9h	1	1	0	1	0	0	0	0		FNOP	None
D9h	1	1	1	0	0	0	0	0		FCHS	ST(0)
D9h	1	1	1	0	0	0	0	1		FABS	ST(0)
D9h	1	1	1	0	0	1	0	0		FTST	ST(0)
D9h	1	1	1	0	0	1	0	1		FXAM	ST(0)
D9h	1	1	1	0	1	0	0	0		FLD1	ST(0), const
D9h	1	1	1	0	1	0	0	1		FLDL2T	ST(0), const
D9h	1	1	1	0	1	0	1	0		FLDL2E	ST(0), const
D9h	1	1	1	0	1	0	1	1		FLDPI	ST(0), const

Table 18. *SuperMathSX Operands (continued)*

T-49-12-05

Opcode/Instruction Encoding										Instruction	Operand(s) (dest,source)
1st Byte	2nd Byte								Bytes 3-7		
B7	B6	B5	B4	B3	B2	B1	B0				
D9h	1	1	1	0	1	1	0	0		FLDLG2	ST(0), const
D9h	1	1	1	0	1	1	0	1		FLDLN2	ST(0), const
D9h	1	1	1	0	1	1	1	0		FLDZ	ST(0), const
D9h	1	1	1	1	0	0	0	0		F2XM1	ST(0)
D9h	1	1	1	1	0	0	0	1		FYL2X	ST(0)
D9h	1	1	1	1	0	0	1	0		FPTAN	ST(0)
D9h	1	1	1	1	0	0	1	1		FPATAN	ST(0)
D9h	1	1	1	1	0	1	0	0		FXTRACT	ST(0)
D9h	1	1	1	1	0	1	0	1		FPREM1	ST(0)
D9h	1	1	1	1	0	1	1	0		FDECSTP	None
D9h	1	1	1	1	0	1	1	1		FINCSTP	None
D9h	1	1	1	1	1	0	0	0		FPREM	ST(0)
D9h	1	1	1	1	1	0	0	1		FYL2XP1	ST(0)
D9h	1	1	1	1	1	0	1	0		FSQRT	ST(0)
D9h	1	1	1	1	1	0	1	1		FSINCOS	ST(0)
D9h	1	1	1	1	1	1	0	0		FRNDINT	ST(0)
D9h	1	1	1	1	1	1	0	1		FSCALE	ST(0)
D9h	1	1	1	1	1	1	1	0		FSIN	ST(0)
D9h	1	1	1	1	1	1	1	1		FCOS	ST(0)
D9h	MD1	MD0	0	0	0	RM2	RM1	RM0	SIB,DISPL	FLD	ST(0), mem32r
D9h	MD1	MD0	0	1	0	RM2	RM1	RM0	SIB,DISPL	FST	mem32r, ST(0)
D9h	MD1	MD0	0	1	1	RM2	RM1	RM0	SIB,DISPL	FSTP	mem32r, ST(0)
D9h	MD1	MD0	1	0	0	RM2	RM1	RM0	SIB,DISPL	FLDENV	mem14/28byte
D9h	MD1	MD0	1	0	1	RM2	RM1	RM0	SIB,DISPL	FLDCW	mem2byte
D9h	MD1	MD0	1	1	0	RM2	RM1	RM0	SIB,DISPL	FSTENV	mem14/28byte
D9h	MD1	MD0	1	1	1	RM2	RM1	RM0		FSTCW	mem2byte
D9Ah	1	1	0	1	1	0	0	1		FUCOMPP	ST(1), ST(0)
DAh	MD1	MD0	0	0	0	RM2	RM1	RM0	SIB,DISPL	FIADD	ST(0), mem32i
DAh	MD1	MD0	0	0	1	RM2	RM1	RM0	SIB,DISPL	FIMUL	ST(0), mem32i
DAh	MD1	MD0	0	1	0	RM2	RM1	RM0	SIB,DISPL	FICOM	ST(0), mem32i
DAh	MD1	MD0	0	1	1	RM2	RM1	RM0	SIB,DISPL	FICOMP	ST(0), mem32i
DAh	MD1	MD0	1	0	0	RM2	RM1	RM0	SIB,DISPL	FISUB	ST(0), mem32i
DAh	MD1	MD0	1	0	1	RM2	RM1	RM0	SIB,DISPL	FISUBR	ST(0), mem32i

Table 18. SuperMathSX Operands (continued)

T-49-12-05

Opcode/Instruction Encoding										Instruction	Operand(s) (dest,source)
1st Byte	2nd Byte								Bytes 3-7		
	B7	B6	B5	B4	B3	B2	B1	B0			
DAh	MD1	MD0	1	1	0	RM2	RM1	RM0	SIB,DISPL	FIDIV	ST(0), mem32i
DAh	MD1	MD0	1	1	1	RM2	RM1	RM0	SIB,DISPL	FIDIVR	ST(0), mem32i
D8h	1	1	1	0	0	0	0	0		See Note 1	
D8h	1	1	1	0	0	0	0	1		See Note 2	
DBh	1	1	1	0	0	0	1	0		FCLEX	None
DBh	1	1	1	0	0	0	1	1		FINIT	None
DBh	1	1	1	0	0	1	0	0		See Note 3	
DBh	MD1	MD0	0	0	0	RM2	RM1	RM0	SIB,DISPL	FILD	ST(0), mem32i
DBh	MD1	MD0	0	1	0	RM2	RM1	RM0	SIB,DISPL	FIST	mem32i, ST(0)
DBh	MD1	MD0	0	1	1	RM2	RM1	RM0	SIB,DISPL	FISTP	mem32i, ST(0)
DBh	MD1	MD0	1	0	1	RM2	RM1	RM0	SIB,DISPL	FLD	ST(0), mem80r
DBh	MD1	MD0	1	1	1	RM2	RM1	RM0	SIB,DISPL	FSTP	ST(n), ST(0)
DCh	1	1	0	0	0	REG2	REG1	REG0		FADD	ST(n), ST(0)
DCh	1	1	0	0	1	REG2	REG1	REG0		FMUL	ST(n), ST(0)
DCh	1	1	1	0	0	REG2	REG1	REG0		FSUBR	ST(n), ST(0)
DCh	1	1	1	0	1	REG2	REG1	REG0		FSUB	ST(n), ST(0)
DCh	1	1	1	1	0	REG2	REG1	REG0		FDIVR	ST(n), ST(0)
DCh	1	1	1	1	1	REG2	REG1	REG0		FDIV	ST(n), ST(0)
DCh	MD1	MD0	0	0	0	RM2	RM1	RM0	SIB,DISPL	FADD	ST(0), mem64r
DCh	MD1	MD0	0	0	1	RM2	RM1	RM0	SIB,DISPL	FMUL	ST(0), mem64r
DCh	MD1	MD0	0	1	0	RM2	RM1	RM0	SIB,DISPL	FCOM	ST(0), mem64r
DCh	MD1	MD0	0	1	1	RM2	RM1	RM0	SIB,DISPL	FCOMP	ST(0), mem64r
DCh	MD1	MD0	1	0	0	RM2	RM1	RM0	SIB,DISPL	FSUB	ST(0), mem64r
DCh	MD1	MD0	1	0	1	RM2	RM1	RM0	SIB,DISPL	FSUBR	ST(0), mem64r
DCh	MD1	MD0	1	1	0	RM2	RM1	RM0	SIB,DISPL	FDIV	ST(0), mem64r

¹ This encoding may be generated by some assemblers. However, the SuperMathSX coprocessor executes an FNOP instruction when it is presented this encoding, maintaining industry standard compatibility. This encoding corresponds to the 8087/287 FENI instruction.

² This encoding may be generated by some assemblers. However, the SuperMathSX coprocessor executes an FNOP instruction when it is presented this encoding, maintaining industry standard compatibility. This encoding corresponds to the 8087/287 FDISI instruction.

³ This encoding may be generated by some assemblers. However, the SuperMathSX coprocessor executes an FNOP instruction when it is presented this encoding, maintaining industry standard compatibility. This encoding corresponds to the 8087/287 FSETPM instruction.

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002714 3T4 ■ CHP

Table 18. SuperMathSX Operands (continued)

T-49-12-05

Opcode/Instruction Encoding										Instruction	Operand(s) (dest,source)
1st Byte	2nd Byte								Bytes 3-7		
	B7	B6	B5	B4	B3	B2	B1	B0			
DCh	MD1	MD0	1	1	1	RM2	RM1	RM0	SIB,DISPL	FDIVR	ST(0), mem64r
DDh	1	1	0	0	0	REG2	REG1	REG0		FFREE	ST(n)
DDh	1	1	0	1	0	REG2	REG1	REG0		FST	ST(n), ST(0)
DDh	1	1	0	1	1	REG2	REG1	REG0		FSTP	mem80r, ST(0)
DDh	1	1	1	0	0	REG2	REG1	REG0		FUCOM	ST(n)
DDh	1	1	1	0	1	REG2	REG1	REG0		FUCOMP	ST(n)
DDh	MD1	MD0	0	0	0	RM2	RM1	RM0	SIB,DISPL	FLD	ST(0), mem64r
DDh	MD1	MD0	0	1	0	RM2	RM1	RM0	SIB,DISPL	FST	mem64r, ST(0)
DDh	MD1	MD0	0	1	1	RM2	RM1	RM0	SIB,DISPL	FSTP	mem64r, ST(0)
DDh	MD1	MD0	1	0	0	RM2	RM1	RM0	SIB,DISPL	FRSTOR	mem94/108byte
DDh	MD1	MD0	1	1	0	RM2	RM1	RM0	SIB,DISPL	FSAV	mem94/108byte
DDh	MD1	MD0	1	1	1	RM2	RM1	RM0		FSTSW	mem2byte
DEh	1	1	0	0	0	REG2	REG1	REG0		FADDP	ST(n), ST(0)
DEh	1	1	0	0	1	REG2	REG1	REG0		FMULP	ST(n), ST(0)
DEh	1	1	0	1	1	0	0	1		FCOMPP	ST(1), ST(0)
DEh	1	1	1	0	0	REG2	REG1	REG0		FSUBR	ST(n), ST(0)
DEh	1	1	1	0	1	REG2	REG1	REG0		FSUBP	ST(n), ST(0)
DEh	1	1	1	1	0	REG2	REG1	REG0		FDIVRP	ST(n), ST(0)
DEh	1	1	1	1	1	REG2	REG1	REG0		FDIVP	ST(n), ST(0)
DEh	MD1	MD0	0	0	0	RM2	RM1	RM0	SIB,DISPL	FIADD	ST(0), mem16i
DEh	MD1	MD0	0	0	1	RM2	RM1	RM0	SIB,DISPL	FIMUL	ST(0), mem16i
DEh	MD1	MD0	0	1	0	RM2	RM1	RM0	SIB,DISPL	FICOM	ST(0), mem16i
DEh	MD1	MD0	0	1	1	RM2	RM1	RM0	SIB,DISPL	FICOMP	ST(0), mem16i
DEh	MD1	MD0	1	0	0	RM2	RM1	RM0	SIB,DISPL	FISUB	ST(0), mem16i
DEh	MD1	MD0	1	0	1	RM2	RM1	RM0	SIB,DISPL	FISUBR	ST(0), mem16i
DEh	MD1	MD0	1	1	0	RM2	RM1	RM0	SIB,DISPL	FIDIV	ST(0), mem16i
DEh	MD1	MD0	1	1	1	RM2	RM1	RM0		FIDIVR	ST(0), mem16i

CHIPS & TECHNOLOGIES INC 51E D 2098116 0002715 230 CHP

Table 18. SuperMathSX Operands (continued)

T-49-12-05

Opcode/Instruction Encoding										Instruction	Operand(s) (dest,source)
1st Byte	2nd Byte										
	B7	B6	B5	B4	B3	B2	B1	B0	Bytes 3-7		
DFh	1	1	1	0	0	0	0	0		FSTSW	80386SX AX Register
DFh	MD1	MD0	0	0	0	RM2	RM1	RM0	SIB,DISPL	FILD	ST(0), mem16i
DFh	MD1	MD0	0	1	0	RM2	RM1	RM0	SIB,DISPL	FIST	mem16i, ST(0)
DFh	MD1	MD0	0	1	1	RM2	RM1	RM0	SIB,DISPL	FISTP	mem16i, ST(0)
DFh	MD1	MD0	1	0	0	RM2	RM1	RM0	SIB,DISPL	FBLD	ST(0), mem80b
DFh	MD1	MD0	1	0	1	RM2	RM1	RM0	SIB,DISPL	FILD	ST(0), mem64i
DFh	MD1	MD0	1	1	0	RM2	RM1	RM0	SIB,DISPL	FBSTP	mem80b, ST(0)
DFh	MD1	MD0	1	1	1	RM2	RM1	RM0	SIB,DISPL	FISTP	mem64i, ST(0)

AC/DC Characteristics

T-49-12-05

The tables and figures in this section describe the operating environment and signal timings required by the SuperMathSX coprocessor. The signal timings correspond to those of 80386SX-compatible CPUs and provide complete interface compatibility.

General Characteristics

Use of the SuperMathSX coprocessor within the ranges specified in Table 19 is mandated for guaranteed operation.

Table 19. *Recommended Operating Conditions*

Symbol	Parameter	Min.	Max.	Unit	Notes
V _{cc}	Supply voltage	4.75	5.25	V	
T _c	Case temperature	0	70	°C	
I _{oh}	Output high current	—	-1.0	mA	V _{oh} = min V _{oh}
I _{ol}	Output low current	—	4.0	mA	V _{ol} = max V _{ol}
I _{ik}	Input clamp current	—	±10	mA	V _{in} V _{ss} or V _{in} V _{cc}
I _{ok}	Output clamp current	—	±25	mA	V _{out} V _{ss} or V _{out} V _{cc}

Table 20 states the most extreme conditions that will be tolerated by the chip without permanent chip damage. Note that these are not continuous operating conditions; the chip will probably be damaged if operated near these limits for a prolonged period of time.

Table 20. *Maximum Tolerated Conditions*

Symbol	Parameter	Min.	Max.	Unit
V _{cc}	Supply voltage	-0.5	6.0	V
	Case temperature (powered)	0	100	°C
	Storage temperature (unpowered)	-65	150	°C
	Voltage on any pin	-0.5	V _{cc} + 0.5	V
	Power dissipation at 5.25V	—	1.9	W

CHIPS & TECHNOLOGIES INC 51E D 2098116 0002717 003 CHP

Table 21 shows input, output, and I/O capacitance values for 5V operation.

T-49-12-05

Table 21. *Capacitance*

Symbol	Parameter	Max.	Unit	Test Condition
C_{in}	Input capacitance	10	pF	$f_c = 1\text{MHz}$
C_{out}	I/O, output capacitance	12	pF	$f_c = 1\text{MHz}$
C_{clk}	Clock capacitance	20	pF	$f_c = 1\text{MHz}$

DC Characteristics

Table 22 provides the DC operating characteristics of the SuperMathSX coprocessor.

Table 22. *DC Characteristics*

Symbol	Parameter	Min.	Max.	Unit	Test Condition
V_{il}	Input low voltage	0	0.8	V	
V_{ih}	Input high voltage	2.0	V_{cc}	V	
V_{cl}	Clock input low voltage	0	0.8	V	
V_{ch}	Clock input high voltage	3.7	V_{cc}	V	
V_{ol}	Output low voltage	—	0.45	V	$I_{ol} = 4.0\text{mA}$
V_{oh}	Output high voltage	2.4	—	V	$I_{oh} = 1.0\text{mA}$
I_{il}	Input leakage current	—	± 15	μA	$V_{in} = 0\text{V to } V_{cc}$
I_{oz}	Output leakage current	—	± 15	μA	$V_{out} = 0\text{V to } V_{cc}$
I_{cc}	V_{cc} supply current	—	380	mA	@ 50MHz (80mA typical)

AC Characteristics

T-49-12-05

Table 23 shows the AC operating characteristics of a 25MHz SuperMathSX coprocessor. The timings listed are with respect to the CPUCLK2 input unless otherwise noted.

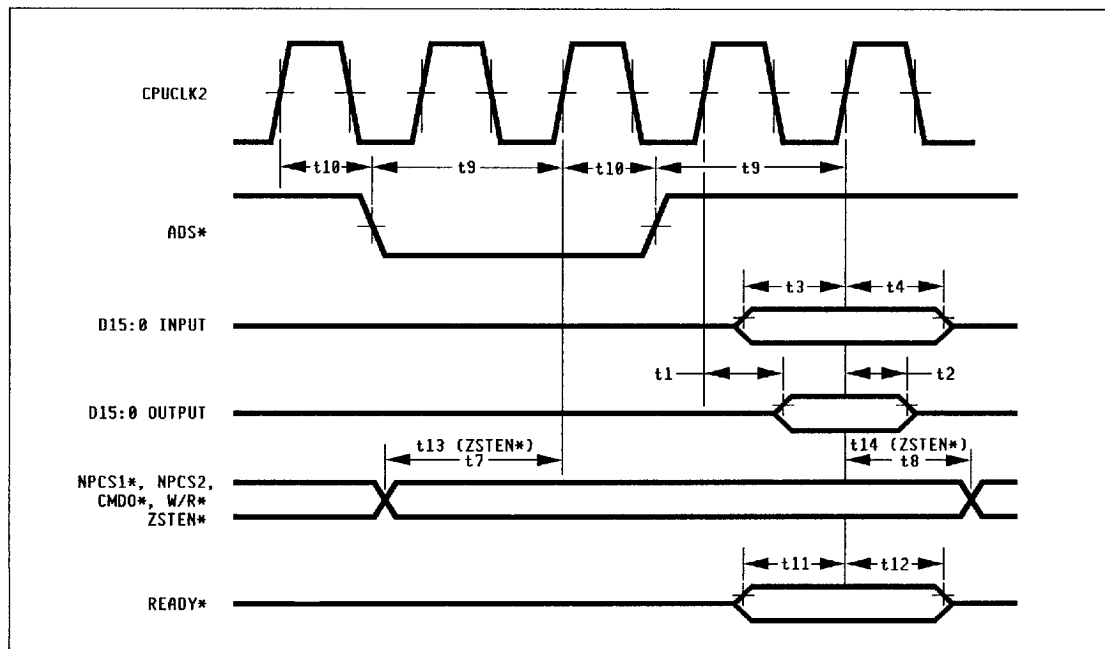
Table 23. AC Operating Characteristics

Symbol	Parameter	Min.	Max.	Unit	Figure Number
Data Bus (D15:0) Timings					
t1	Data output delay ($C_L = 50\text{pF}$)	0	27	ns	25
t2	Data output float time ($C_L = 50\text{pF}$)	5	24	ns	25
t3	Data input setup time	8		ns	25
t4	Data input hold time	11		ns	25
Output Signal (READY*, PEREQ, BUSY*, ERROR*) Timings					
t5	READY* output delay ($C_L = 50\text{pF}$)	3	19	ns	27
t5	PEREQ output delay ($C_L = 50\text{pF}$)	4	24	ns	27
t5	BUSY* output delay ($C_L = 50\text{pF}$)	4	24	ns	27
t5	ERROR* output delay ($C_L = 50\text{pF}$)	4	24	ns	27
t6	Float time from ZSTEN* active/delay time from ZSTEN* inactive	1	30	ns	27
Control Signal (CMD0*, NPCS1*, NPCS2, W/R*) Timings					
t7	Setup time	12		ns	25
t8	Hold time	0		ns	25
Other Signal Timings					
t9	ADS* setup time	14		ns	25
t10	ADS* hold time	4		ns	25
t11	READY* setup time	8		ns	25
t12	READY* hold time	4		ns	25
t13	ZSTEN* setup time	12		ns	25
t14	ZSTEN* hold time	2		ns	25
t15	RESETIN setup time	10		ns	26
t16	RESETIN hold time	3		ns	26
t17	RESETIN duration	40		CPUCLK2s	26
t18	RESETIN inactive before first opcode write	50		CPUCLK2s	26
Clock Input (CPUCLK2) Timings					
t19	Period	20	500	ns	29
t20	Time above 2.0V	6.25		ns	29

Table 23. AC Operating Characteristics (continued)

T-49-12-05

Symbol	Parameter	Min.	Max.	Unit	Figure Number
Clock Input (CPUCLK2) Timings (continued)					
t21	Time above min V_{ch}	4.5		ns	29
t22	Time below 2.0V	6.25		ns	29
t23	Time below 0.8V	4.5		ns	29
t24	Fall time from min V_{ch} to 0.8V		6	ns	29
t25	Rise time from 0.8V to min V_{ch}		6	ns	29
Control Interface Relative Timings					
t26	BUSY* duration	16		CPUCLK2s	28
t27	ERROR* active to BUSY* inactive	6		CPUCLK2s	28
t28	PEREQ inactive to ERROR* active	6		CPUCLK2s	28
t29	READY* active to BUSY* active	4	4	CPUCLK2s	28
t30	Time from opcode write to next write (opcode or operand)	6		CPUCLK2s	28
t31	Time from operand write to next operand write	8		CPUCLK2s	28

Figure 25. Data I/O Timings

CHIPS & TECHNOLOGIES INC 51E D ■ 2098116 0002720 6TB ■ CHP

Figure 26. Reset Timings

T-49-12-05

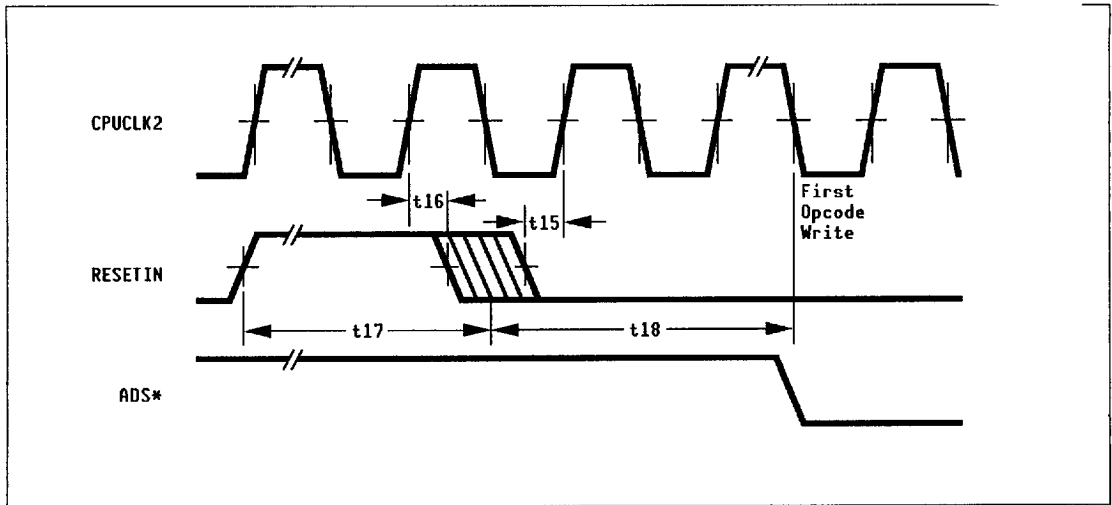


Figure 27. Timing After Change of ZSTEN* State

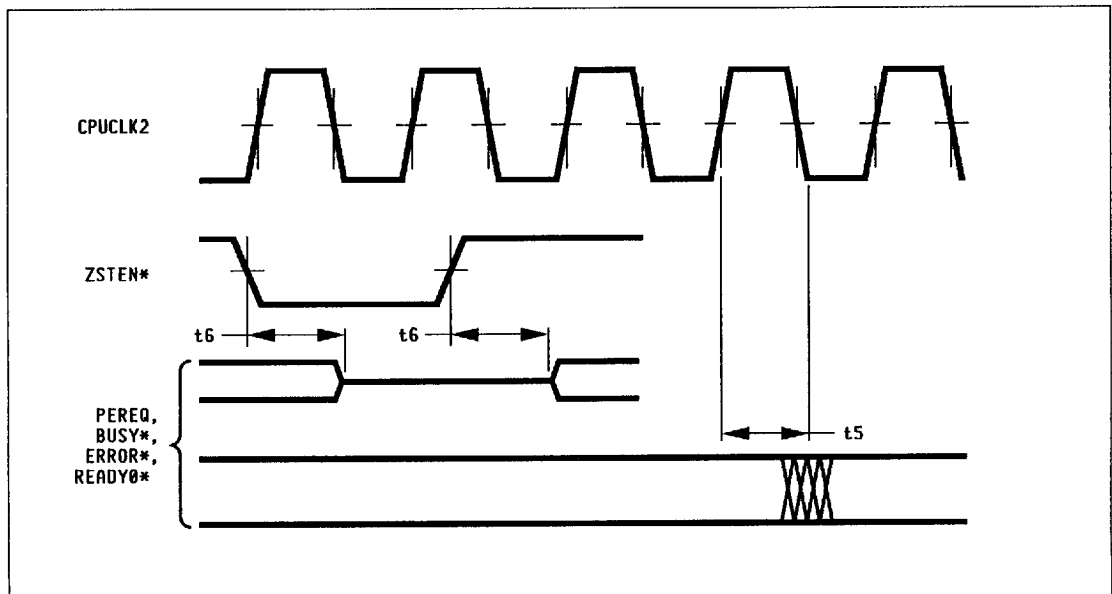
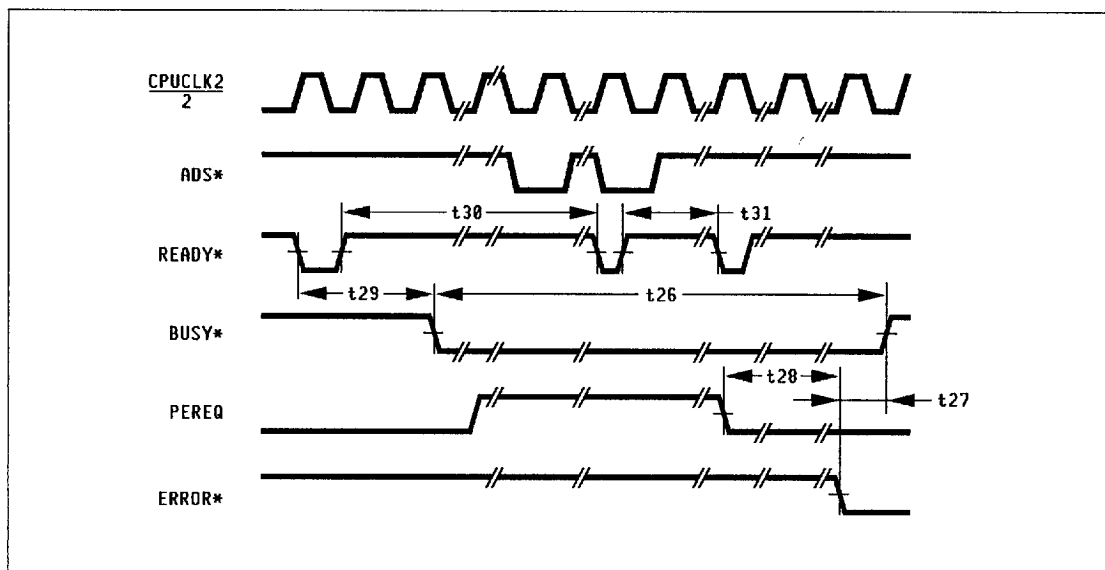
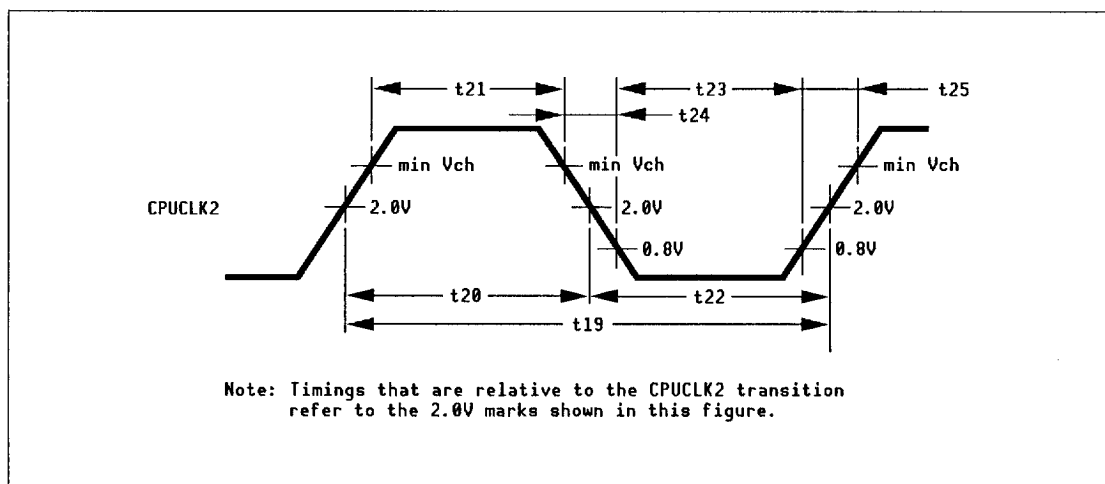


Figure 28. Relative Control Signal Timings

T-49-12-05

**Figure 29.** CPUCLK2 Waveform Characteristics

Packaging Dimensions

T-49-12-05

Figure 30 shows the dimensions of the 68-pin PLCC package.

Figure 30. Plastic Leaded Chip Carrier (PLCC)